UCRL-TR-210372

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Phase Sensitive Cueing for 3D Objects in Overhead Images

David W. Paglieroni

10 March 2005

Auspices Statement

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Phase Sensitive Cueing for 3D Objects in Overhead Images

David W. Paglieroni
Lawrence Livermore National Laboratory
P.O. Box 808, L-290, Livermore, CA 94550

20 January 2005

1. Introduction

Locating specific 3D objects in overhead images is an important problem in many remote sensing applications. 3D objects may contain either one connected component or multiple disconnected components. Solutions must accommodate images acquired with diverse sensors at various times of the day, in various seasons of the year, or under various weather conditions. Moreover, the physical manifestation of a 3D object with fixed physical dimensions in an overhead image is highly dependent on object physical dimensions, object position/orientation, image spatial resolution, and imaging geometry (e.g., obliqueness).

This paper describes a two-stage computer-assisted approach for locating 3D objects in overhead images. In the matching stage, the computer matches models of 3D objects to overhead images. The strongest degree of match over all object orientations is computed at each pixel. Unambiguous local maxima in the degree of match as a function of pixel location are then found. In the cueing stage, the computer sorts image thumbnails in descending order of figure-of-merit and presents them to human analysts for visual inspection and interpretation. The figure-of-merit associated with an image thumbnail is computed from the degrees of match to a 3D object model associated with unambiguous local maxima that lie within the thumbnail. This form of computer assistance is invaluable when most of the relevant thumbnails are highly ranked, and the amount of inspection time needed is much less for the highly ranked thumbnails than for images as a whole.

1.1 Overview of Model Matching

Section 2 describes how to project surfaces of solid models for 3D objects onto overhead images. It is argued that visible edges associated with projected surfaces are appropriate and often sufficient for model matching because unlike other pixels, model edges are characteristically salient and visible in overhead images, independent of the imaging sensor and acquisition conditions.

A robust measure for matching projected model edges to overhead images is developed in Section 4. It is based on *pixel phase*. For monochrome and multi-band images, the phase of a pixel refers to the direction of flow from light to dark at that pixel. For images of projected model edges, the phase of an edge pixel refers to the direction of the normal to the edge boundary at that pixel. As such, pixel phase is an angle measured in the image plane from a pixel. Algorithms for estimating phase for projected edge pixels and pixels in monochrome or multi-band images are developed in Section 3.

Matching techniques based on pixel phase seek consistency in pixel phase between pairs of images. Phase sensitive matching is relatively insensitive to variations in image source because it deals primarily with directional (i.e., phase, as opposed to amplitude) information at the pixel level. It is somewhat related to a multi-source image auto-registration technique known as normalized cross-correlation of complex gradients (NCCCG), except NCCCG uses both amplitude and phase information from the gradient in pixel amplitude to match images acquired by different sensors (Eppler2000, Eppler2003). Phase sensitive matching is more closely related to the matching technique based on pixel phase in Eppler2001 that automatically geo-registers images by matching images to projections of 3D site model edges. This paper seeks to extend phase sensitive matching to the problem of matching projected model edges for 3D objects to overhead images.

Phase sensitive matching can be placed into the context of other techniques for matching projected model edges to images by categorizing such techniques as those based on edges, those based primarily on pixel amplitude (luminance), and those based primarily on pixel phase (direction or orientation). Edge matching techniques require edges to first be extracted from images (Prewitt1970, Davis1975, Marr1980, Canny1986) so that projected model edges can be matched to them. Chamfer matchers compute average distances from image edge pixels to the nearest projected model edge pixels or vice-versa (Barrow1977). Chamfer matchers are based on distance transforms of edge maps, i.e., arrays of distances from each pixel to the nearest edge pixel (Rosenfeld1966, Danielson1980, Yamada1984, Borgefors1988, Paglieroni1992). When applied to object detection, edge matching techniques are known to be highly sensitive to edge clutter and the quality of detected edges (Paglieroni1994).

Since one of the images is a map of projected model edges and the other is monochrome or multi-band, techniques that attempt to match pixel amplitude require the monochrome or multi-band images to first be subjected to edge-enhancement pre-processing so that edges end up bright relative to the background. Normalized cross-correlation is a form of cross-correlation that is relatively insensitive to bias and gain in pixel amplitude (Kumar1992, Brown1992, Fonseca1996). A related method based on correlation coefficients (which vary from −1 to 1) has properties similar to normalized cross-correlation (Svedlow1976). Phase correlators use the phase of the DFT of the spatial correlation of two images (Kuglin1975, DeCastro1987). The translational offset between two images is chosen as the location of the maximum of the inverse DFT of the phase image. Matching techniques based primarily on pixel amplitude tend to work well when the two images being matched are comparable, as in the case of two images of an area acquired with the same sensor at nearly the same time. However, the often fail when the two images are highly disparate, as in the case of two images of an area acquired with different sensors, or an image and a line map of the same area (Eppler2000).

Model matchers typically generate match surfaces that contain one match value per pixel. It is usually necessary to select certain points on the match surface as candidate locations of objects, and these points typically correspond to local maxima in the match surface. Match disambiguation techniques that distinguish between ambiguous and unambiguous local maxima are discussed in Section 5. In this paper, cueing results are based solely on the unambiguous local maxima.

As discussed previously, the 3D object matching system developed in Sections 2-5 scales from monochrome to multi-band images. It also scales to images of arbitrary size because it processes images block-by-block. Any image can be decomposed into a series of potentially overlapping image blocks of fixed or variable size. As discussed in Section 6, block overlap is used to capture objects that span neighboring blocks, and block size variation is used to accommodate variations in spatial resolution across an image.

1.2 Assisted vs. Automated Image Analysis and Interpretation

The goals of the computer-automated and computer-assisted approaches to analysis and interpretation of overhead images are fundamentally different, both philosophically and in practice. This paper strongly advocates the computer-assisted approach (see Section 7). Although computers can be programmed to quickly extract low-level features from large volumes of imagery, they often cannot easily be programmed to consistently interpret this information correctly. Conversely, human analysts are good at correctly interpreting images, but cannot quickly handle large volumes of imagery. The computer-automated approach attempts to replace strong human abilities to analyze and interpret images with computer programs that are presently immature by comparison. In contrast, the computer-assisted approach uses strengths of computers to enhance strengths of human analysts. In the case of 3D objects, the computer-automated approach requires the computer to detect and even recognize objects (a problematic task by today's standards). In contrast, the computer-assisted approach only requires the computer to cue human analysts, who then have the responsibility to recognize, analyze and interpret what they see.

2. Object Spatial Models

Object spatial models can exist either in image space (model dimensions are in pixels) or object space (model dimensions are in meters). If the goal is robust object cueing, it makes sense to limit object spatial models to edge points that are characteristically visible in overhead images, independent of the imaging sensor and acquisition conditions. In this case, the spatial patterns that get matched to the image are projections of model edges onto the image. For image space models, the projections are typically established by manually drawing edges with a mouse over an instance of the object in an image and rotating the resulting pattern in the image plane. For object space models, the projections are established by using rational polynomials to project model edges onto the image plane at each of several different model orientations, as described later in this section.

Image space models are strictly 2D. They apply when spatial resolution and effective viewing angle are nearly fixed across all pixels in all the images to be analyzed, and the viewing angle is fairly close to nadir, as in the case of ortho-rectified images with fixed ground sample distance. On the other hand, object space models can be either 2D or 3D, and are applicable to any combination of images. They can be represented as collections of point, line, curve, shape and volumetric graphics primitives. Such primitives can be represented parametrically. For example, polygons can be represented with sequences of vertex coordinates, whereas cones and cylinders can each be represented with two points and one radius. Various commercial standard formats can be used to express solid models for objects in terms of such primitives.

2.1 Forward Projection

Forward projection is the process of mapping object space coordinates [*x,y,z*] to image space or pixel [*column,row*] coordinates [*c,r*]. Pixel coordinates [*c,r*] are traditionally expressed as *rational polynomials* [*C(x,y,z), R(x,y,z)*] of object space coordinates. These rational polynomials are of the form

$$(2.1)\qquad f(x,y,z) = \sum_{(i,j,k)\in N_f} a_f(i,j,k)\, x^i y^j z^k \Big/ \sum_{(i,j,k)\in D_f} b_f(i,j,k)\, x^i y^j z^k$$

where N_f and D_f are sets of distinct non-negative integer-valued exponents for the numerator and denominator polynomials. The headers of geo-registered images either explicitly contain the rational polynomial coefficients a_f and b_f , or parameters from which rational polynomial coefficients can be derived.

Within the local vicinity of a specific pixel in an overhead image, rational polynomial expressions for pixel [*column,row*] coordinates [*c,r*] can be approximated linearly using *affine models*, i.e., first-order linear combinations of object space coordinates [*x,y,z*]:

$$(2.2)\qquad \begin{aligned} c &= C(x,y,z) \approx a_{00}x + a_{01}y + a_{02}z + a_{03} \\ r &= R(x,y,z) \approx a_{10}x + a_{11}y + a_{12}z + a_{13} \end{aligned}$$

Affine models can be used to project spatially localized 3D objects onto specific locations within an overhead image. By using affine models as opposed to higher-order rational polynomials, the computational cost of forward projection can be significantly reduced. Two local linearization techniques for rational polynomials are given below:

Taylor Series Linearization

In first-order Taylor series linearization, *f(x,y,z)* is approximated as

(2.3) $$f(x,y,z) \approx f(x_0,y_0,z_0) + [\, x - x_0 \,, y - y_0 \,, z - z_0 \,] \cdot \left. \begin{bmatrix} \partial f(x,y,z) / \partial x \\ \partial f(x,y,z) / \partial y \\ \partial f(x,y,z) / \partial z \end{bmatrix} \right|_{[x,y,z] = [x_0,y_0,z_0]}$$

for $[x,y,z]$ in the vicinity of $[x_0, y_0, z_0]$. Although partial derivatives for rational polynomials in equation (2.3) can be derived analytically, numerical estimates are sometimes more convenient:

(2.4a) $$\left. \frac{\partial f\,(x,y,z)}{\partial x} \right|_{[x,y,z] = [x_0, y_0, z_0]} \approx \frac{f\,(x_0 + \delta, y_0, z_0) - f\,(x_0 - \delta, y_0, z_0)}{2\delta}$$

(2.4b) $$\left. \frac{\partial f\,(x,y,z)}{\partial y} \right|_{[x,y,z] = [x_0, y_0, z_0]} \approx \frac{f\,(x_0, y_0 + \delta, z_0) - f\,(x_0, y_0 - \delta, z_0)}{2\delta}$$

(2.4c) $$\left. \frac{\partial f\,(x,y,z)}{\partial z} \right|_{[x,y,z] = [x_0, y_0, z_0]} \approx \frac{f\,(x_0, y_0, z_0 + \delta\,) - f\,(x_0, y_0, z_0 - \delta\,)}{2\delta}$$

In equations (2.4), δ is a small increment (say $\delta = 1$ meter).

Least-Squares Linearization

In least-squares linearization, rational polynomials $f(x,y,z)$ are evaluated at $n \geq 4$ *contrived* points $\boldsymbol{p_k} = [x_k, y_k, z_k]$, $k = 1 \ldots n$ in the vicinity of $[x_0, y_0, z_0]$. The exact coordinates of these four points are not important. However, they must be relatively close to $[x_0, y_0, z_0]$, and at least four of them must be linearly independent. As an example, if $n = 4$ and δ is some relatively small spatial offset, four contrived points that satisfy these restrictions are

(2.5) $$\begin{aligned} \boldsymbol{p_1} &= [\, x_0 - \delta, y_0 - \delta, z_0 - \delta/10 \,] & \qquad \boldsymbol{p_2} &= [\, x_0 + \delta, y_0 - \delta, z_0 + \delta/10 \,] \\ \boldsymbol{p_3} &= [\, x_0 + \delta, y_0 + \delta, z_0 - \delta/10 \,] & \qquad \boldsymbol{p_4} &= [\, x_0 - \delta, y_0 + \delta, z_0 + \delta/10 \,] \end{aligned}$$

The coefficients in equation (2.2) are obtained by finding the least-squares solution to the system of $n \geq 4$ linear equations $[c_k, r_k] \overset{\Delta}{=} [C(x_k,y_k,z_k), R(x_k,y_k,z_k)] = [a_{00}x_k + a_{01}y_k + a_{02}z_k + a_{03}, a_{10}x_k + a_{11}y_k + a_{12}z_k + a_{13}]$ for $k = 1 \ldots n$, where $C(x,y,z)$ and $R(x,y,z)$ are rational polynomials:

(2.6) $$\begin{bmatrix} a_{00} & a_{10} \\ a_{01} & a_{11} \\ a_{02} & a_{12} \\ a_{03} & a_{13} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ . & . & . & . \\ x_n & y_n & z_n & 1 \end{bmatrix}^{-} \cdot \begin{bmatrix} c_1 & r_1 \\ . & . \\ c_n & r_n \end{bmatrix}$$

In equation (2.6), "–" represents the pseudo-inverse of an nx4 matrix, where $n \geq 4$ (for $n = 4$, the matrix pseudo-inverse is a matrix inverse).

2.2 Geopositioning

Geopositioning (also sometimes referred to as *inverse projection*) is the process of mapping pixel [*column,row*] coordinates [c,r] to object space coordinates [x,y,z]. Monoscopic geopositioning (which uses one image, as opposed to stereoscopic geopositioning, which uses two images) requires a 3D terrain model for the area of interest. Geopositioning is normally accomplished by computing the line-of-sight through object space associated with pixel [c,r], and then finding the point where the line-of-sight first intersects the surface. If the surface model is simple (e.g., constant elevation terrain), it may be possible to find the point of intersection analytically. Otherwise, ray tracing may be necessary.

Although rational polynomials are normally used for forward projection, they can also be used for geopositioning if the terrain has constant elevation (z is fixed). In particular, it is possible to use rational polynomial coefficients to estimate geo-coordinates [x_0, y_0] at an assumed height $z = z_0$ for a pixel [c_0, r_0] by solving the equations

$$(2.7) \qquad [C(x,y,z_0), R(x,y,z_0)] = [c_0, r_0]$$

analytically for [x,y]. Analytical solutions can be readily derived for certain rational polynomials of less than order 3. Otherwise, analytical solutions can be very difficult to derive and numerical solutions are indicated.

Newton-Like Methods

A numerical method for computing the solution to equations (2.7) using Newton-like iterations is given below (Grant2004):

```
$k \leftarrow 1, \; [x_1, y_1] \leftarrow [\mu_x, \mu_y]$
while $[C(x_k, y_k, z_0) - c_0]^2 + [R(x_k, y_k, z_0) - r_0]^2 \geq \varepsilon$
    Find $[x_{k+1}, y_{k+1}]$ such that
        (2.8) $\begin{bmatrix} c_0 \\ r_0 \end{bmatrix} = \begin{bmatrix} C(x_k, y_k, z_0) \\ R(x_k, y_k, z_0) \end{bmatrix} + \left.\begin{bmatrix} \frac{\partial C(x,y,z_0)}{\partial x} & \frac{\partial C(x,y,z_0)}{\partial y} \\ \frac{\partial R(x,y,z_0)}{\partial x} & \frac{\partial R(x,y,z_0)}{\partial y} \end{bmatrix}\right|_{[x,y]=[x_k,y_k]} \cdot \begin{bmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{bmatrix}$
    $k \leftarrow k+1$
$[x_0, y_0] \leftarrow [x_k, y_k]$
```

An initial value [μ_x, μ_y] for [x, y] is required. [μ_x, μ_y] is typically supplied with the rational polynomials as an [x, y] "offset" for the image that lies somewhere in or near the field-of-view. Although partial derivatives for rational polynomials can be derived analytically, numerical estimates are sometimes more convenient. The numerical estimates in equations (2.4) for the partial derivatives require two rational polynomial evaluations each, so our Newton-like method requires approximately *10N* rational polynomial evaluations if it converges in N iterations. The algorithm can be expected to converge quickly if $C(x,y,z_0)$ and $R(x,y,z_0)$ have no local peaks or valleys for [x, y] within an area that contains [μ_x, μ_y] and extends somewhat beyond image field-of-view (this should be a valid assumption).

When the algorithm converges, the computed geo-coordinates $[x_0, y_0, z_0]$ are guaranteed to project to pixel coordinates that are within $\sqrt{\varepsilon}$ pixels of $[c_0, r_0]$. By choosing $\varepsilon \leq 1$, the computed geo-coordinates can be derived to a projection accuracy of within one pixel. By choosing a significantly larger value for ε (say $\varepsilon = 100$ pixels), the number of iterations can be reduced, but the projected geo-coordinates are only guaranteed to be fairly close to the correct pixel coordinates $[c_0, r_0]$. Moreover, if the initial value $[\mu_x, \mu_y]$ for $[x, y]$ is already fairly close to the correct value for $[x, y]$ at the assumed height $z = z_0$, it may be possible to bypass the loop altogether. Also, if a larger value of ε is used, the approximate geo-coordinates $[x_0, y_0, z_0]$ output from the Newton-like iterations can be used as a point in the region-of-interest for the true solution, which can be derived using linearization methods, as described below.

Linearization Methods

Linearization methods compute precise estimates for geo-coordinates $[x_0, y_0, z_0]$ from rough estimates. Local linearizations of rational polynomials in the vicinity of a rough geo-coordinate estimate are derived using either Taylor series or least-squares methods (see Section 2.1). Locally linearized versions of rational polynomials (equation (2.1)) have affine representations (equation (2.2)). The geo-coordinates of the block center at an assumed height of $z = z_0$ can be computed from the pixel coordinates $[c_0, r_0]$ and affine model coefficients as

(2.9)
$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}^{-1} \cdot \begin{bmatrix} c_0 - a_{02} z_0 - a_{03} \\ r_0 - a_{12} z_0 - a_{13} \end{bmatrix}$$

Rough geo-coordinate estimates can be obtained by applying Newton-like iterations (as in the previous section) with a sufficiently large value of ε. They can also be obtained from rough approximations of the $[x,y]$ coordinates for the four corners of the image chip. Given approximate coordinates $[C_k, R_k, X_k, Y_k]$ $k = 1 \ldots 4$ for the four corners of the image chip, the least-squares fit coefficients in the equations

(2.10)
$$\begin{aligned} x &\approx \alpha_{00} c + \alpha_{01} r + \alpha_{02} \\ y &\approx \alpha_{10} c + \alpha_{11} r + \alpha_{12} \end{aligned}$$

are computed as

(2.11)
$$\begin{bmatrix} \alpha_{00} & \alpha_{10} \\ \alpha_{01} & \alpha_{11} \\ \alpha_{02} & \alpha_{12} \end{bmatrix} = \begin{bmatrix} \sum C_k^2 & \sum C_k R_k & \sum C_k \\ \sum C_k R_k & \sum R_k^2 & \sum R_k \\ \sum C_k & \sum R_k & 4 \end{bmatrix}^{-1} \cdot \begin{bmatrix} C_1 & C_2 & C_3 & C_4 \\ R_1 & R_2 & R_3 & R_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ X_3 & Y_3 \\ X_4 & Y_4 \end{bmatrix}$$

Rough geo-coordinates $\boldsymbol{p}_R = [x_R, y_R, z_0]$ at an assumed height $z = z_0$ for an image pixel $[c_0, r_0]$ can be estimated directly from these six α coefficients and the image rational polynomials. $[x_R, y_R, z_0]$ is required to lie within Δ meters in x and y of the true geo-coordinates $[x_0, y_0, z_0]$ (say $\Delta = 300\text{m}$). An initial estimate $[x', y'$ for $[x_R, y_R]$ is computed as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ r_0 \end{bmatrix} + \begin{bmatrix} \alpha_{02} \\ \alpha_{12} \end{bmatrix} \tag{2.12}$$

Then

$$[x_R, y_R] \leftarrow \underset{[x,y] \in R(x',y',\Delta)}{\arg\min} \; [C(x, y, z_0) - c_0]^2 + [R(x, y, z_0) - r_0]^2 \tag{2.13}$$

where $R(x',y',\Delta) = \{[x, y] : x = x' - n\Delta \ldots x' + n\Delta$ and $y = y' - n\Delta \ldots y' + n\Delta$ in increments of $\Delta\}$. Choose $n = 2$ initially. If $x_R = x' \pm n\Delta$ or $y_R = y' \pm n\Delta$ (i.e., the local minimum is on the border of the search region), then $n \leftarrow 2n$ and re-compute $[x_R, y_R]$ using equation (2.13). Else, $\boldsymbol{p}_R \leftarrow [x_R, y_R, z_0]$. This process requires 25 rational polynomial evaluations if the search range never needs to be doubled.

2.3 Projected Model Edges

Image space models are composed of edge pixels corresponding to a single object orientation. Object space models are composed of point, line, polyline, curve (e.g., spline curve), shape (polygon, ellipse, etc.), and volumetric graphics primitives (boxes, pyramids, spheres, cylinders, cones, etc.). Parametric representations of graphics primitives can be stored in a commercial standard format so that they can be passed between different systems. A point can be represented as an ordered pair. A line can be represented by a slope and intercept. Polylines and polygons can be represented by sequences of vertex ordered pairs. A curve can be represented by a sequence of control points. An ellipse can be represented by a centroid and major/minor axis lengths. Boxes and pyramids can be represented by ordered sets of vertex coordinates in 3D. A sphere can be represented by centroid coordinates in 3D and a radius. Cylinders and cones can be represented by a line segment in 3D and a radius. In addition, the surfaces of graphics primitives may have textures that are important to capture. Textural properties of surfaces are modeled separately from shape and volumetric properties of surfaces.

Projections of object space model edges are formed by projecting each model surface onto the image (using established computer graphics techniques) and keeping only the surface edges. As each new surface is projected, edges from portions of previously projected surfaces that become occluded (hidden) are removed.

There are two types of *surface edges*. *Silhouette edges* correspond to boundaries of projected surfaces against background imagery or other projected surfaces. These consist of planar surface edges and edges of regions in the image occluded by non-planar surfaces. For example, a tall cylindrical smoke stack consists of silhouette edges along the borders of the planar horizontal circle surfaces at the top and bottom, as well as linear vertical boundaries along the silhouette of the curved cylindrical surface against the background. *Texture edges* correspond to texture patterns present on surfaces (such as painted letters). These can be texture-mapped onto each surface immediately after it has been projected.

Flattened projections are produced from flattened object space models, i.e., object space models in which z is forced to be constant for all parameters of all graphics primitives that the 3D solid model is composed of. Flattened projections are useful for establishing overlap between candidate object detections (see Section 5). Note that flattened projections have no hidden surfaces or lines.

Projections of an object space model for a single object can vary with object position and orientation, and from image to image. However, projections at fixed orientation will be virtually invariant to changes in object position within sufficiently small image blocks (and often across many connected image blocks). Therefore, when an image is processed block-by-block (to accommodate images of arbitrary size), no more than one set of rotated model projections corresponding to one object position within image block field-of-view needs to be generated for each image block. These projections are generated subject to an assumed terrain elevation $z = z_0$ typically stored in the image header, which can be quite inaccurate, especially over areas of high terrain relief. However, when collection platform altitude is very much greater than errors in assumed terrain elevation, the errors will have negligible effect on the model projections.

For a given image block, projected model edges are generated in three steps. In step 1, the model is rotated about the z axis through the model centroid in some angular increment (discussed in Section 4). This step only needs to be performed once per object. In step 2, the rotated models are translationally offset such that the model centroid is shifted to the geo-coordinates of the pixel at the center of the image block (see Section 2.2). In step 3, the translated and rotated models are projected onto the image block, retaining only the edges. Hidden surfaces and edges are removed using established computer graphics techniques. Steps 2-3 are performed once per image block unless it can be assumed that the projections do not vary significantly from block to block.

2.4 Projection Size Estimation

Because image chips are often too large to be stored in computer memory, they must be processed block-by-block. As discussed in Section 6, image block size and overlap are determined from the extents of object model projections. Although such extents can be easily derived directly from projections, it can be considerably more expensive to generate projections over multiple object positions and orientations in the first place. Moreover, the projections themselves are not even required for block size and overlap estimation.

If one does not expect to encounter tilted objects, the size or extent of an object projection can be quickly estimated from a bounding cylinder formed by rotating the object about the vertical (z) axis through its centroid. The bounding cylinder centroid is the object centroid. The height h of the cylinder is the height of the object model, and the cylinder radius R is the xy distance from the object model centroid to the point on the object model that is most remote in xy distance. The extent of the projection is somewhat over-estimated by the distance (in pixels) from the projection of the centroid to the pixel on the projection of the circular top of the cylinder that it is farthest from.

Mathematically, suppose the centroid of an object model is to be projected onto an image pixel $[c_0, r_0]$. xy geo-coordinates $[x_0, y_0]$ at an assumed height of $z = z_0$ can be derived for this pixel as described in Section 2.2. Within the vicinity of $[x_0, y_0, z_0]$, the affine representation in equation (2.2) for the rational polynomials that express pixel coordinates $[c, r]$ in terms of object space coordinates $[x, y, z]$ can be derived using Taylor series or least-squares linearization (see Section 2.1). Now the locus of points in object space that lie on the edge of the circular top of the cylinder is the set of points $[x, y] = [x_0+R\cos\theta, y_0+R\sin\theta, z_0+h/2]$ for $0 \le \theta < 2\pi$. From equation (2.2), the distance d (in pixels) from $[c_0, r_0]$ to the projection $[c, r]$ of a point on the edge of the circular top of the cylinder is given by

$$\begin{aligned} d^2 &= (c - c_0)^2 + (r - r_0)^2 \\ &= [a_{00}R\cos\theta + a_{01}R\sin\theta + a_{02}h/2]^2 + [a_{10}R\cos\theta + a_{11}R\sin\theta + a_{12}h/2]^2 \end{aligned} \tag{2.14}$$

An estimate of the extent of the object projection is found as the maximum of d in equation (2.14) with respect to θ.

3. Phase Estimation for Pixels

The matching measure developed in this paper is based primarily on the phase (i.e., the angle or direction) of directional derivatives in pixel intensity. Matching measures based on pixel phase are likely to be robust across images acquired by different sensors under diverse conditions because they either utilize pixel amplitude information to a very limited extent or ignore it altogether. Techniques for estimating pixel phase for both monochrome/multi-band images and projected model edges are derived in this section.

3.1 Pixel Phase in Monochrome and Multi-Band Images

For monochrome or multi-band images u, the directional derivative $\dot{u}(c,r)$ at pixel $[c,r]$ can be estimated as a weighted sum of direction vectors from the center of pixel $[c,r]$ to the centers of neighbor pixels $[c',r']$. $\dot{u}(c,r)$ can be expressed as a complex number with amplitude $A(c,r)$ and *pixel phase* (i.e., angle or direction) $\theta(c,r)$.

$R(c,r \mid \rho)$ is the *neighborhood* of pixel $[c,r]$. It contains all pixels, exclusive of $[c,r]$, that intersect the circle with *neighborhood radius* $\rho = 1, 2 \ldots$ centered on pixel $[c,r]$:

(3.1) $$R(c,r \mid \rho) \overset{\Delta}{=} \{ [c',r'] \neq [c,r] : (|c' - c| - \tfrac{1}{2})^2 + (|r' - r| - \tfrac{1}{2})^2 < \rho^2 \}$$

$\dot{u}(c,r)$ can be expressed in the general form

(3.2) $$\dot{u}(c,r) \overset{\Delta}{=} A(c,r)\, e^{j\theta(c,r)} = \sum_{(c',r') \in R(c,r \mid \rho)} [\, u(c',r') - u(c,r) \,] \cdot \frac{(c'-c) + j(r'-r)}{[(c'-c)^2 + (r'-r)^2]^{k/2}}$$

The directional derivative estimate $\dot{u}(c,r)$ has two parameters: the neighborhood radius ρ (see equation (3.1)) and the *distance power* $k = 0, 1 \ldots$ (see equation (3.2)). The distance power is the exponent on Euclidean distance from the center of pixel $[c,r]$ to the center of pixel $[c',r']$. Note that $[\rho, k] = [1,0]$ corresponds to the Prewitt directional derivative estimate (Prewitt1970), and $[\rho, k] = [1,2]$ corresponds to a scaled Sobel directional derivative estimate (Davis1975).

If $k = 0$ in equation (3.2), the direction vector $(c'-c) + j(r'-r)$ is uniformly weighted, so for a fixed value of $u(c',r') - u(c,r)$, neighbor pixels $[c',r']$ farther from pixel $[c,r]$ have a greater impact on $\dot{u}(c,r)$ when they should have less. If $k = 1$, the direction vector is a unit vector, so for a fixed value of $u(c',r') - u(c,r)$, neighbor pixels $[c',r']$ have the same impact on $\dot{u}(c,r)$ regardless of their distance from $[c,r]$. If $k = 2$ and $u(c',r') - u(c,r)$ is fixed, the impact that individual neighbor pixels $[c',r']$ have on $\dot{u}(c,r)$ varies inversely with distance from $[c,r]$, but collectively, the set of all pixels at fixed distance from $[c,r]$ have the same impact on $\dot{u}(c,r)$, independent of distance. However, if $k = 3$ and $u(c',r') - u(c,r)$ is fixed, the set of all pixels at fixed distance from $[c,r]$ collectively have an impact on $\dot{u}(c,r)$ that varies inversely with distance. We therefore use $k = 3$.

The signal-to-noise ratio (SNR) of the phase estimate can be expected to increase as the neighborhood radius ρ increases, but the computational complexity of phase estimation will also increase. A small value of ρ should be used if the patterns contain finely spaced detail, whereas larger values should be used if the patterns contain mostly coarse level detail. We use $\rho = 1$ for image pixel phase estimation when matching object projections. A pixel phase example is shown in Fig.3.1 for a football field image (courtesy of TerraServerUSA).

Equation (3.2) applies directly to monochrome images u, but it can easily be generalized to multi-band images with B bands by writing equation (3.2) for each band, summing the B equations into a single composite equation, and dividing both sides of the composite equation by B. This is tantamount to interpreting $u(c,r)$ in equation (3.2) to be the mean of values of pixel $[c,r]$ across all B bands. For multi-band images u, $u(c,r)$ in equation (3.2) should thus be interpreted as the value of pixel $[c,r]$ in a band-averaged version of u.

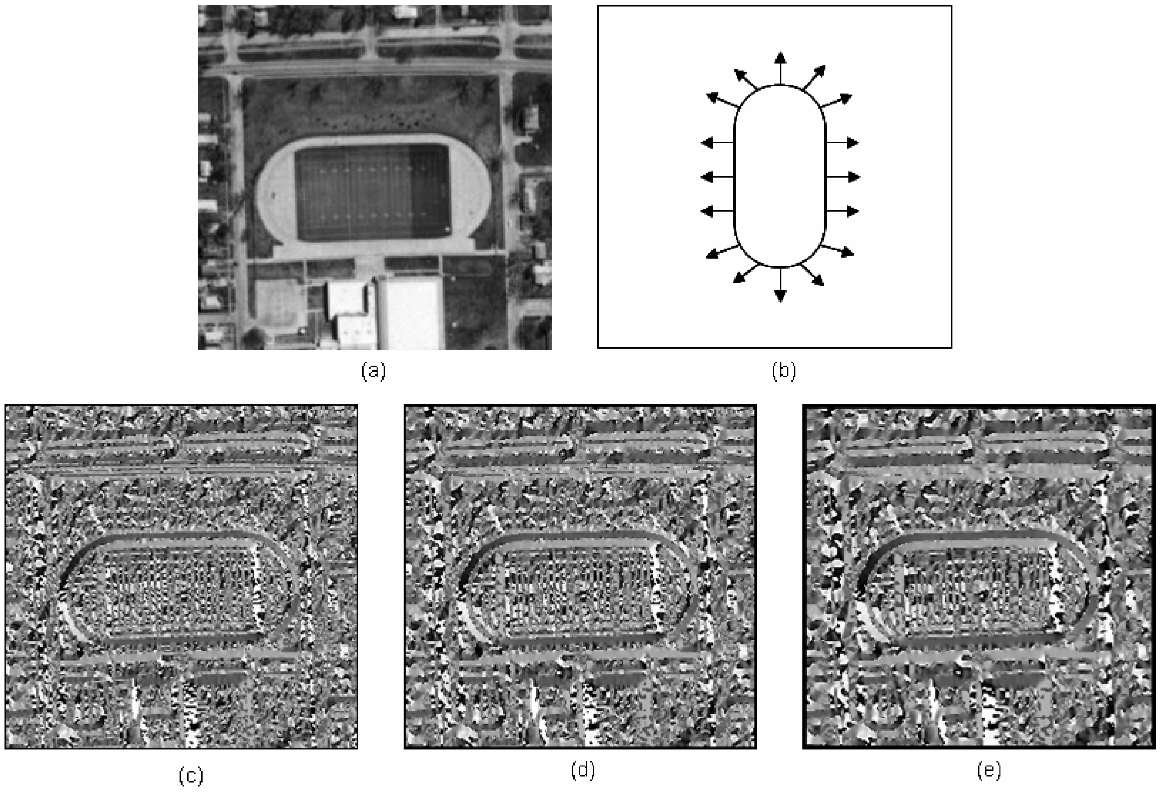


Fig.3.1 (a) Image of football field. (b) Boundary phases for projected model edges of football field. (c)-(e) Pixel phases for image of football field for k = 3 and ρ = 1, 2 and 3.

3.2 Boundary Phase for Edge and Curve Pixels

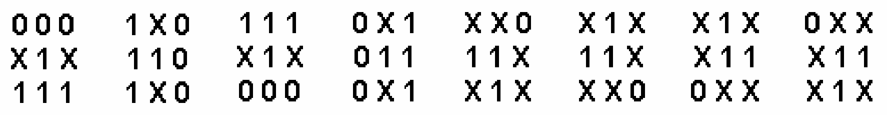
From the standpoint of pixel phase, edges and curves are treated differently than monochrome and multi-band images. In practice, the phase of an edge or curve pixel is not computed as the angle of a directional derivative in pixel intensity. It is instead computed geometrically as the angle of the normal to the projected edge or curve boundary at that pixel.

Image Space Model Projections

For a curve pixel $[c,r]$ in an image space model projection, let $\Omega(c,r|\rho)$ be the set of pixels within the square neighborhood of width $2\rho+1$ centered on $[c,r]$ that lies on the set of 8-connected pixels that contains $[c,r]$. The *boundary phase* $\beta(c,r)$ at curve pixel $[c,r]$ can be estimated as a weighted average of angles $\phi([c',r'], [c,r])$ of rays from the center of pixel $[c,r]$ to the centers of neighbor pixels $[c',r'] \in \Omega(c,r|\rho)$. The weights should be non-negative and sum to one. Let us design a phase estimator for curve pixels $[c,r]$ in which the impact that neighbor pixel $[c',r']$ has on $\beta(c,r)$ decreases linearly with distance from $[c,r]$ to $[c',r']$:

$$\beta(c,r) = \frac{\displaystyle\sum_{(c',r')\in\Omega(c,r\,|\,\rho)} \frac{\phi([c',r'],[c,r])}{[(c'-c)^2+(r'-r)^2]^{1/2}}}{\displaystyle\sum_{(c',r')\in\Omega(c,r\,|\,\rho)} \frac{1}{[(c'-c)^2+(r'-r)^2]^{1/2}}} \tag{3.3}$$

In equation (3.3), it is assumed that the curves have been completely thinned. Structuring elements for complete thinning are shown in Fig.3.2. Equation (3.3) also only applies to curve pixels $[c,r]$ that are 8-connected to at least one other curve pixel (so the denominator cannot be zero). A small value of ρ should be used if the curve patterns contain finely spaced detail, whereas larger values should be used if the curve patterns contain mostly coarse level detail. We use $\rho = 1$ for boundary pixel phase estimation when matching object projections. A boundary phase example is shown in Fig.3.1.



```
000   1X0   111   0X1   XX0   X1X   X1X   0XX
X1X   110   X1X   011   11X   11X   X11   X11
111   1X0   000   0X1   X1X   XX0   0XX   X1X
```

Fig.3.2 Morphological structuring elements for complete iterative curve thinning. They represent patterns to be detected in the curve map. "X" represents a "don't care" state (either "0" or "1"). When a structuring element pattern is detected, the curve pixel beneath the center of the 3x3 neighborhood is set to zero.

Equation (3.3) will not generally produce a coherent (unambiguous) estimate of the phase at curve pixel $[c,r]$ if $[c,r]$ is the endpoint of a curve, or $[c,r]$ belongs to more than one curve (i.e., it lies at the junction of two or more curve segments). For $\rho = 1$, curve pixel $[c,r]$ is *boundary phase coherent* if and only if there are exactly two curve pixels in its 8-neighborhood. Let $\dot{R}(c,r|\rho)$ be the set of pixels on the border of the square of half-width ρ centered on $[c,r]$. For $\rho > 1$, curve pixel $[c,r]$ is phase coherent if and only if there are exactly two curve pixels in its 8-neighborhood and for $k = 1 \ldots \rho - 1$, each curve pixel 8-connected to $[c,r]$ in $\dot{R}(c,r|k)$ has exactly two curve pixels in its 8-neighborhood, exactly one of which lies in $\dot{R}(c,r|k+1)$. Since boundary phase estimates computed using equation (3.3) are only valid for boundary phase coherent curve pixels, it is important to remove phase incoherent curve pixels before computing the phase match between an image and projected model edges.

Object Space Model Projections

One could use the same method to compute boundary phases for both image and object space model projections. However in practice, boundary phases should be computed differently for object space model projections. The problem is that for sufficiently low image spatial resolutions and certain orientations of certain object space models, one can expect significant percentages of projected edge pixels to be boundary phase incoherent. Since phase incoherent pixels must be removed from edge projections prior to phase matching, one may be forced to use severely trimmed projection patterns, and this can significantly reduce the quality of the matches.

This problem can be mostly avoided by estimating boundary phases as the model surfaces and edges are being projected, rather than after all the curves and edges have been projected and merged (as in the case of image space models). As each curve is projected, it is possible to estimate the boundary phase (either analytically or by using equation (3.3)) of each pixel that is not an endpoint or junction point. For example, each pixel on a projected line segment has the same phase, namely the angle of the normal to the line segment, which can be computed analytically from the projected endpoints.

4. Phase Sensitive Matching

The degree of phase sensitive match between an image and projected model edges is based on the disparity between boundary phase over all projected edge pixels and pixel phase at the corresponding image pixels. Phase sensitive matching is intended for images acquired by potentially different sensors under potentially diverse conditions. Since it is based on relative values of neighboring pixel amplitudes, phase sensitive matching is much less sensitive to brightness differences inherent in multi-source images of the same area than matching techniques based directly on pixel amplitude. When it takes only edge pixels into account, phase sensitive matching results are based solely on pixels more likely to be salient, regardless of image source, which should promote consistency in matching results across images from different sources. In this case, phase sensitive matching is primarily concerned with correlation in edge direction, as opposed to edge strength, so one might expect it to be relatively insensitive to contrast or even texture variations between images from different sources.

Phase sensitive matching is primarily intended for detecting distinctive objects. An object can be distinctive due to its size, shape, or surface details. For example, a small rectangular object will not be distinctive in an image that contains large numbers of small regions that are nearly rectangular, but a large rectangular object might be very distinctive. However, a small rectangular object may be distinctive if it has unusual surface markings. Also, an object with an unusual shape might be distinctive even it is relatively small.

4.1 FFT-Based Phase Sensitive Matching

Consider an image $\boldsymbol{\theta} \overset{\Delta}{=} \{\theta(c,r) \;\; c = 0\ldots w{-}1;\; r = 0\ldots h{-}1\}$ of pixel phases for pixels in a monochrome or multi-band image block with w columns and h rows of pixels (see equation (3.2)). Let $\boldsymbol{P}$ be a list of $N_P > 0$ boundary phase coherent projected model edge pixels for some object projected onto the image block at some orientation. Consider a second image $\boldsymbol{\beta} \overset{\Delta}{=} \{\beta(c,r) \;\; c,r = 0\ldots 2R_{max}\}$ of boundary phases for the bitmap of projected model edges associated with $\boldsymbol{P}$ (see equation (3.3)). The *model projection radius* R_{max} is the radius of the projection bounding circle, i.e., the distance, in pixels, from the projected edge centroid to the most remote projected edge pixel rounded up to the nearest integer.

Next, for pixels $[c,r]$ in the image block, define amplitude parameters $a(c,r)$ as $a(c,r) = 1$ for $c = 0\ldots w{-}1;\; r = 0\ldots h{-}1$ and $a(c,r) = 0$ otherwise. The impact of brightness differences in diverse images of the same object on phase sensitive matching results is minimized by choosing $a(c,r)$ to be constant (in this case, unity) for all image block pixels. However, it is important to set $a(c,r) = 0$ at pixels $[c,r]$ for which the directional derivative amplitude $A(c,r)$ in equation (3.2) is not greater than A_{min}. This forces the matcher to take into account only image block pixels at which there is readily perceptible contrast variation. For $\rho = 1$ and $k = 3$ in equation (3.2), $A_{min} = 16$ is a reasonable choice (this choice applies if the pixel values are integers and there are 8 or more bits per pixel, as windowed 8-bit ranges in images with more than 8 bits per pixel are viewed and displayed the same way as 8 bit images).

Also, for the N_P boundary phase coherent curve pixels $[c,r] \in \boldsymbol{P}$ on the projected model edges, define non-negative weighting parameters $b(c,r)$ that sum to unity over $[c,r] \in \boldsymbol{P}$. Larger weights are assigned to projected edge pixels that are more important to the match. By default, all projected edge pixels contribute equally to the match, in which case the weighting is uniform, i.e., $b(c,r) = 1 / N_P$ for all $[c,r] \in \boldsymbol{P}$. Techniques for computing non-uniform weights are a topic for future research.

Let us require the similarity $S(\Delta_c,\Delta_r)$ between phase image $\boldsymbol{\beta}$ at an offset of $[\Delta_c,\Delta_r]$ and phase image $\boldsymbol{\theta}$ to vary from 0 (for poor similarity) to 1 (for perfect similarity). Such a measure of similarity can be derived by noticing that the square of the cosine of the difference between two angles β and θ varies from 0 (for angles of vectors pointing in orthogonal directions) to 1 (for angles of vectors pointing in the same or opposite directions). A simple similarity measure that satisfies the design requirements is

(4.1)
$$S(\Delta_c,\Delta_r) = \frac{1}{2} + \sum_{(c,r)\in \boldsymbol{P}} a(c+\Delta_c,r+\Delta_r)\, b(c,r) \left[\cos^2 [\theta(c+\Delta_c,r+\Delta_r) - \beta(c,r)] - \frac{1}{2} \right]$$

$$= \frac{1}{2} + \frac{1}{2} \sum_{(c,r)\in \boldsymbol{P}} a(c+\Delta_c,r+\Delta_r)\, b(c,r) \cos 2[\theta(c+\Delta_c,r+\Delta_r) - \beta(c,r)]$$

Of course, the design requirements are still met even if both factors of "*1/2* " in equation (4.1) are removed. However, the similarity measure in equation (4.1) is preferred because, as shown below, it can be evaluated with a single 2D convolution, whereas two 2D convolutions are required if both factors of "*1/2* " are removed from equation (4.1).

In equation (4.1), it is understood that the dimensions of $\boldsymbol{\theta}$ (namely w and h) must be at least as large as the width of $\boldsymbol{\beta}$ (namely $2R_{max}+1$). The range of admissible offsets is thus Δ_c = 0... $w-2R_{max}-1$ and Δ_r = 0...$h-2R_{max}-1$ (w, $h > 2R_{max}$). Note that an offset of $\boldsymbol{\beta}$ by $[\Delta_c,\Delta_r]$ corresponds to a projection centroid location of $[\Delta_c+R_{max}, \Delta_r+R_{max}]$ in the image block.

In the spatial domain, the computational complexity associated with evaluating $S(\Delta_c,\Delta_r)$ over all projection positions and orientations within an image block is directly proportional to the number of edge pixels (N_P), and can be large even for modest N_P. Fortunately, $S(\Delta_c,\Delta_r)$ can be efficiently evaluated over all projection positions and orientations using the FFT. Since most FFT implementations are radix 2, let us zero-pad the phase images $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ to a width of W columns and a height of H rows of pixels, where W and H are the first powers of 2 ≥ w and h. To maximize computational efficiency, the image block dimensions w and h can be deliberately specified by the user as powers of 2 (in which case $W = w$ and $H = h$). Now consider two complex zero-padded images of size H x W:

(4.2a)
$$\boldsymbol{\Theta} \triangleq \{\Theta(c,r) = a(c,r)\, e^{j2\theta(c,r)} \quad c = 0 \ldots W{-}1;\ r = 0 \ldots H{-}1\}$$

(4.2b)
$$\boldsymbol{B} \triangleq \{B(c,r) = b(c,r)\, e^{j2\beta(c,r)} \quad c = 0 \ldots W{-}1;\ r = 0 \ldots H{-}1\}$$

The block geometry is depicted graphically in Fig.4.1.

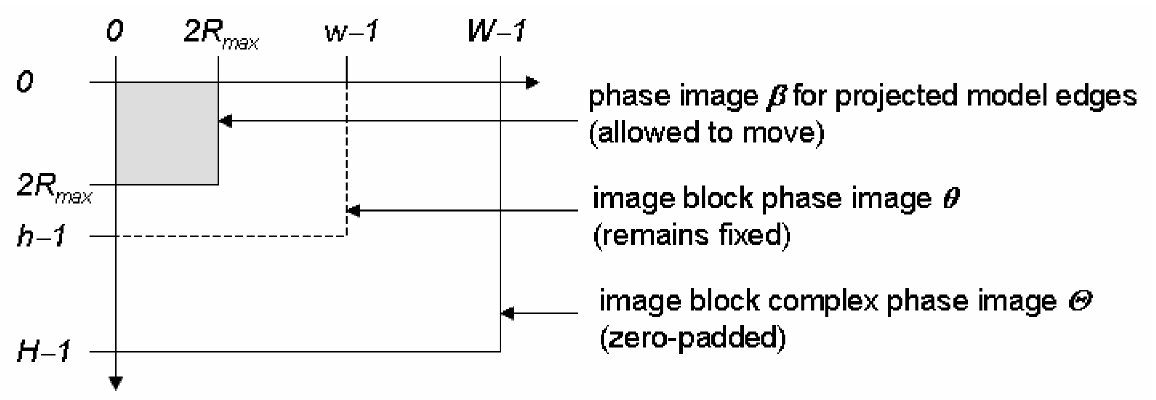

0
2R_max
w–1
W–1
0
2R_max
h–1
H–1
phase image β for projected model edges (allowed to move)
image block phase image θ (remains fixed)
image block complex phase image Θ (zero-padded)

Fig.4.1 Geometry for FFT-based phase sensitive matching.

From equations (4.1)-(4.2),

(4.3) $$S(\Delta_c,\Delta_r) = \frac{1}{2}+\frac{1}{2}Re\left[\sum_{(c,r)\in \boldsymbol{P}} a(c+\Delta_c,r+\Delta_r)\, b(c,r)\, e^{j2[\theta(c+\Delta_c,r+\Delta_r)-\beta(c,r)]}\right]$$

$$= \frac{1}{2}+\frac{1}{2}Re\left[\sum_{c=0}^{W-1}\sum_{r=0}^{H-1} b(c,r)\, e^{-j2\beta(c,r)}\, a(c+\Delta_c,r+\Delta_r)\, e^{j2\theta(c+\Delta_c,r+\Delta_r)}\right]$$

$$= \frac{1}{2}+\frac{1}{2}Re\left[B^*(\Delta_c,\Delta_r) \circledast \Theta(\Delta_c,\Delta_r)\right]$$

$$= \frac{1}{2}+\frac{1}{2}Re\left[B^*(-\Delta_c,-\Delta_r) * \Theta(\Delta_c,\Delta_r)\right]$$

where "$*$" and "$\circledast$" are the 2D spatial convolution and correlation operators:

(4.4a) $$f(\Delta_c,\Delta_r) * g(\Delta_c,\Delta_r) \overset{\Delta}{=} \sum_{c=0}^{W-1}\sum_{r=0}^{H-1} f(c,r)\, g(\Delta_c-c,\Delta_r-r)$$

(4.4b) $$f(\Delta_c,\Delta_r) \circledast g(\Delta_c,\Delta_r) \overset{\Delta}{=} \sum_{c=0}^{W-1}\sum_{r=0}^{H-1} f(c,r)\, g(c+\Delta_c,r+\Delta_r)$$

In equation (4.3), it is assumed that B and Θ from equation (4.2) are periodic with period W in the c dimension and H in the r dimension (i.e., they are periodic [W,H]). If f and g are periodic [W,H], then

(4.5) $$f(\Delta_c,\Delta_r) \circledast g(\Delta_c,\Delta_r) = f(-\Delta_c,-\Delta_r) * g(\Delta_c,\Delta_r)$$

From equation (4.3) and the circular convolution theorem of the DFT,

(4.6) $$S(\Delta_c,\Delta_r) = \frac{1}{2}+\frac{1}{2WH}Re\left[\text{IDFT}\,[\,\text{DFT}\,[B^*(-\Delta_c,-\Delta_r)]\cdot \text{DFT}\,[\Theta(\Delta_c,\Delta_r)]\,]\right]$$

where "DFT" and "IDFT" denote the un-scaled forward and inverse two-sided (2D) discrete Fourier transforms. In matrix form,

(4.7) $$\boldsymbol{S} = \frac{\boldsymbol{1}}{2}+\frac{1}{2WH}\cdot Re\left[\boldsymbol{F}_{\boldsymbol{H}}^{*}\,[\,(\boldsymbol{F}_{\boldsymbol{H}}\,\tilde{\boldsymbol{B}}\,\boldsymbol{F}_{\boldsymbol{W}})\cdot(\boldsymbol{F}_{\boldsymbol{H}}\,\boldsymbol{\Theta}\,\boldsymbol{F}_{\boldsymbol{W}})]\,\boldsymbol{F}_{\boldsymbol{W}}^{*}\right]$$

where $\boldsymbol{S}$ is an H x W matrix of match similarities, $\boldsymbol{1}$ is an H x W matrix of 1's, $\tilde{\boldsymbol{B}}$ is the complex conjugate of $\boldsymbol{B}$ in equation (4.2b) (which is periodic [W,H]) folded about both the rows and columns axis, $\boldsymbol{F}_{\boldsymbol{n}} \overset{\Delta}{=} \{e^{-j2\pi ik/n}\ \ i,k = 0\ldots n-1\}$ is the n x n un-scaled DFT matrix, and $\boldsymbol{A}\cdot\boldsymbol{B} \neq \boldsymbol{AB}$ is the term-wise product (as opposed to the matrix product $\boldsymbol{AB}$) of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Note that the inverse of a unitary DFT matrix is given by its complex conjugate ($\boldsymbol{F}_{\boldsymbol{n}}^{-1} = \boldsymbol{F}_{\boldsymbol{n}}^{*}$), so IDFT matrices are complex conjugates of DFT matrices. Matrices $\boldsymbol{S}$ generated using equation (4.7) contain elements $S(\Delta_c,\Delta_r)$ that are valid only for integers $0 \le \Delta_c < w - 2R_{max}$ and $0 \le \Delta_r < h - 2R_{max}$ (these are the admissible offsets in the discussion following equation (4.1)). Equations (4.6) and (4.7) can be efficiently evaluated using the FFT.

Equations (4.6)-(4.7) mathematically characterize FFT-based phase matching between an image block and an object at one orientation. In general, matching must be performed for each of N object orientations (see Section 4.2) to produce matrices of match similarities $\boldsymbol{S_n}$ $n = 1 \ldots N$. Matrices $\tilde{\boldsymbol{B}}_{\boldsymbol{n}}$ are generated from projected model edges at each of N object orientations $n = 1 \ldots N$. For each admissible offset $[\Delta_c, \Delta_r]$,

$$(4.8) \qquad S(\Delta_c,\Delta_r) = \max_{n = 1 \ldots N} S_n(\Delta_c,\Delta_r)$$

and the angle $\phi(\Delta_c,\Delta_r)$ of best match is saved. Fig.4.2 gives a block diagram for FFT-based phase matching across all orientations.

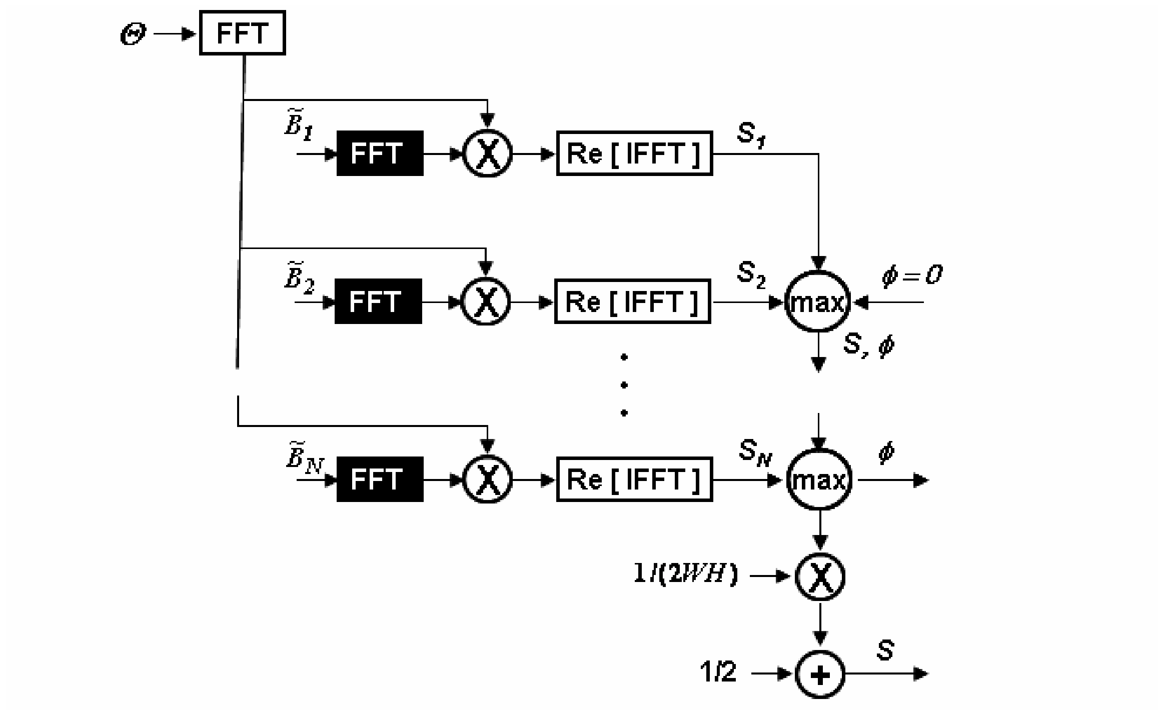


Fig.4.2 Block diagram for FFT-based phase sensitive matching. All DFT's and IDFT's are unscaled. The shaded boxes correspond to computations that may not need to be repeated from image block to image block.

4.2 The Number of Object Orientations and Image Spatial Resolution

At a given position and orientation, the size, shape and general appearance of an object in an image is accurately predicted by the projection of the object onto the image at that position and orientation (see Section 2). When using phase sensitive matching to detect objects at arbitrary position and orientation in an image, one must match the object at a number of orientations. The number of required orientations (N) tends to increase with the size or extent of the projection.

Also, the spatial resolution needed for finding a particular object in an image may be less than the spatial resolution of the image. It is advisable to reduce the spatial resolution of the image as much as possible without compromising detection performance. This provides two significant reductions in computational complexity. First, it takes no more time to process a larger area of coverage in a reduced resolution image than to process a smaller area in a full resolution image. Also, fewer object orientations need to be matched at lower spatial resolution because the object projections will be smaller.

Consider projections of model edges onto an image block at a set of sampled object orientations (say every 10°). Let R_{max} be the radius of the bounding circle for the set of projections. N does not need to exceed the circumference $2\pi R_{max}$ (in pixels) of the bounding circle. Next, define the *weighted mean radius* $\bar{R}$ to be the weighted mean distance from the centroid to the projected model edges over all sampled object orientations, where the weights $b(c,r)$ assigned to projected edge pixels $[c,r]$ in the measure of phase sensitive match (equation (4.1)) are used. It is more appropriate to specify N as directly proportional to $\bar{R}$ as opposed to R_{max} if $\bar{R}$ is significantly less than R_{max}. By choosing $N = int\,(2\pi\bar{R}/k + 0.5)$, the projected edge pixels move, on weighted average, roughly k pixels between successive rotations. If the phase estimates are derived from pixels within 3x3 windows (as for the case of $\rho = 1$ in equations (3.2)-(3.3)), then it makes sense to choose $k = 2$, in which case, $N = int\,(\pi\bar{R} + 0.5)$ would be appropriate.

$\bar{R}$ can be viewed as a simple estimate of projection size or extent (in pixels). By adjusting the spatial resolution of the image such that the mean projection radius matches a designated target value of $\bar{R}_T$, the number of projection orientations to use for object matching (N) can be driven to a value of

$$N = int\,(\pi\bar{R}_T + 0.5) \tag{4.9a}$$

that is constant across all blocks within all images. $\bar{R}_T$ should be chosen as small as possible, but not so small that matching performance will be compromised. For example, $\bar{R}_T = 20$ in equation (4.9) yields $N = 63$, which corresponds to a 5.7° angular increment. If it turns out that R_{max} is less than $\bar{R}_T$, it will not be possible to force the mean projection radius to match the target projection radius unless the image is zoomed-in. If zooming-in is prohibited, then the projection will have to remain at a spatial resolution of 1X, in which case, the required number of projection orientations would be $N = int\,(\pi R_{max} + 0.5)$. Combining this result with equation (4.9a) yields

$$N = int\,[\pi\,min\,(\bar{R}_T, R_{max}) + 0.5] \tag{4.9b}$$

If zooming-in is prohibited, the image block spatial scale factor α is given by

$$\alpha = min\,(1, \bar{R}_T/\bar{R}) \tag{4.10}$$

where the image spatial resolution is αX (e.g., 1/2X). Let w_α be the width (in pixels) of the square image block at a spatial resolution of αX to process. This is the width of the square image blocks to compute 2D DFT's for in phase sensitive matching ($w_\alpha = W = H$ in equations (4.2)-(4.3)), and it should be chosen as some fixed power of 2 (say $w_\alpha = 512$). The width w of the corresponding square image block at 1X spatial resolution is

$$w = int\,(w_\alpha/\alpha + 0.5) \tag{4.11}$$

An image can be spatially zoomed using, for example, bicubic interpolation. Image space model edge projections can be spatially zoomed-out by multiplying the edge pixel coordinates by $\alpha \le 1$, rounding the results to the nearest integer, rasterizing the zoomed edge pixels, completely thinning the resulting bitmap, and reading the thinned edge pixel coordinates. Edge projections for object space models can be spatially zoomed by projecting the model directly onto a zoomed image grid.

4.3 Complexity Analysis and Memory Requirements

The computational complexity per pixel associated with FFT-based phase matching varies in roughly direct proportion to the number of object orientations (N) and the square of the image block spatial scale factor α. Direct computation of the 2D DFT of an HxW image requires $O(WH^2+HW^2)$ complex sums-of-products (CSOP's), whereas only $O(WH\,log_2H+HW\,log_2W) = O(WH\,log_2WH)$ CSOP's are required when a radix 2 FFT is used. Note that one CSOP is equivalent to four operations (OP's), where an OP is defined as a real sum-of-product (one real add and one real multiply). Also, from Fig.4.2, FFT-based phase sensitive matching requires as many as $2N+1$ complex 2D DFT's per image block. However, DFT's for bitmaps of projected model edges do not have to be re-computed from block to block unless the projected model edges change significantly. If none of the N projections change significantly from block to block, only $N+1$ complex 2D DFT's need to be computed. In this case, FFT-based phase matching requires only

(4.12) $$n = O[4NWH\,log_2(WH)] \text{ OP's / block} = O[8NW^2\,log_2W\,] \text{ OP's / block for } H = W$$

The number of operations per pixel depends not only on the number of operations per image block, but also on the image block spatial scale factor α. If W and H are powers of 2 and there is no zero-padding, then the number of operations per pixel is

(4.13) $$n_p = n / \left(\frac{W}{\alpha} \cdot \frac{H}{\alpha} \right) = \alpha^2\, n / (WH)$$
$$= O[4\alpha^2 N\,log_2(WH)] \text{ OP's / pixel} = O[8\alpha^2 N\,log_2W\,] \text{ OP's / pixel for } H = W$$

For typical values of N, H and W (say $N = 100$ and $H = W = 512$) and the worstcase value of unity for α, the computational cost is $O(1.9 \times 10^9)$ OP's / block and $n_p = O(7200)$ OP's / pixel. Clearly, parallel processing may be needed to achieve a prescribed pixel throughput rate. However, at least the computational cost does not depend on the number of edge pixels when the FFT is used.

The realization of phase sensitive matching in Fig.4.2 requires enough memory for $N+2$ DFT's of size HxW. One would normally perform the FFT computations in double precision and store the resulting DFT elements in single precision (4 bytes each). In this case, the RAM size (B) in bytes must satisfy the inequality

(4.14) $$B > 8(N+2)WH$$

For example, if $N = 50$ and $H = W = 512$, then a RAM size of at least 110 MByte is needed. If $N = 50$, 1 GByte of RAM is available, and the blocks are square, then the maximum admissible block width that is also a power of 2 is $H = W = 1024$.

4.4 Examples

Fig.4.3 shows a 512x512 section of an image of a prison in Calipatria, CA (courtesy of TerraServerUSA). Fig.4.4 shows image space model edge projections to be matched against the prison image (a "notched" building, a long building, and a "grated" building). Fig.4.5 shows surfaces ***S*** of match similarities for the models of the three types of buildings in Fig.4.4 against the prison image in Fig.4.3. The similarities range from 0 (dark) to 1 (light). Each model was matched at the image block spatial scale factor α specified in equation (4.10), and these scale factors are reflected in the sizes of the resulting match surfaces. For $\bar{R}_T = 20$, the values of α computed for the notched, long and grated buildings were 0.769, 0.513 and 1.0. Note that from equation (4.9b), $\bar{R}_T = 20$ and $R_{max} \geq \bar{R}_T \Rightarrow N = 63$ model orientations for matching (a 5.7° angular increment). The black regions surrounding the match surfaces are a result of the fact that valid similarities only occur within a range of offsets $[\Delta_c, \Delta_r]$ that shrinks as the model projection radius increases (see the discussion following equation (4.1)).

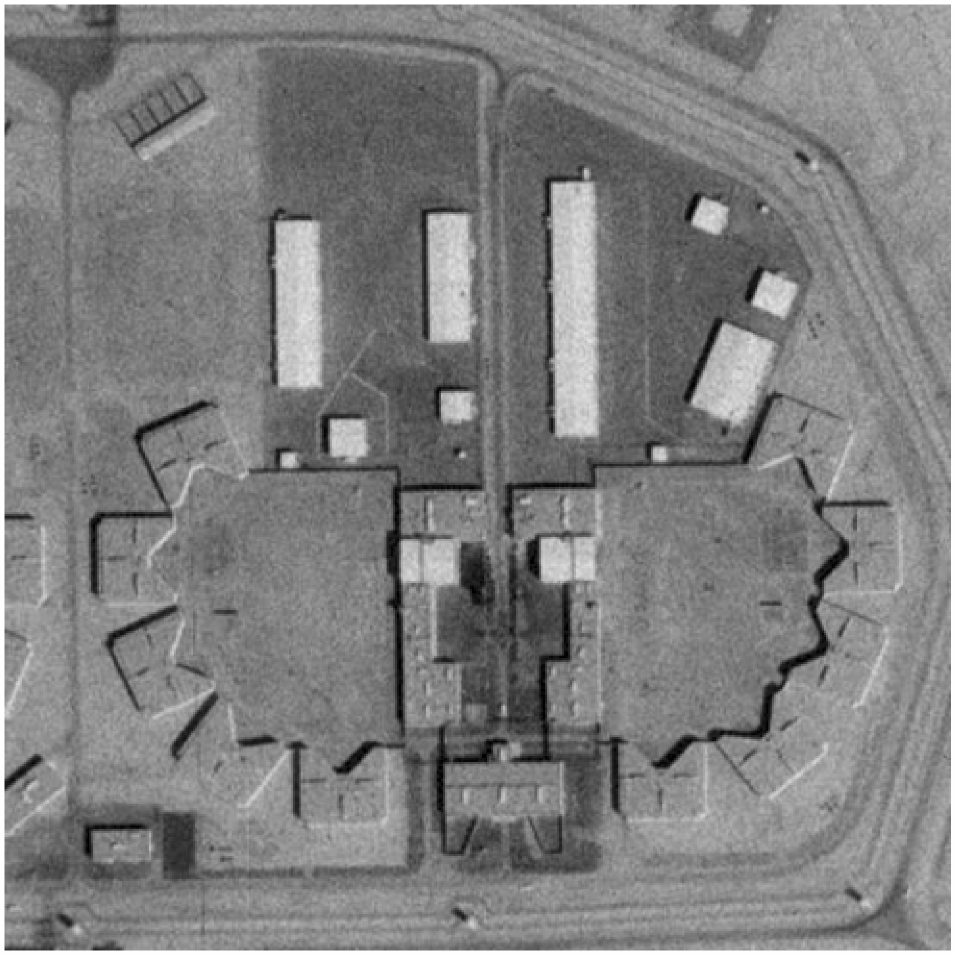


Fig.4.3 512x512 section of prison image for matching experiments.

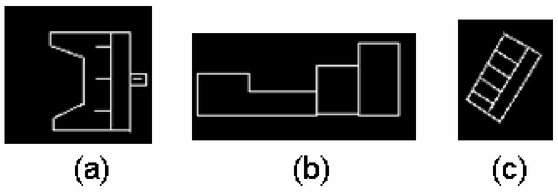
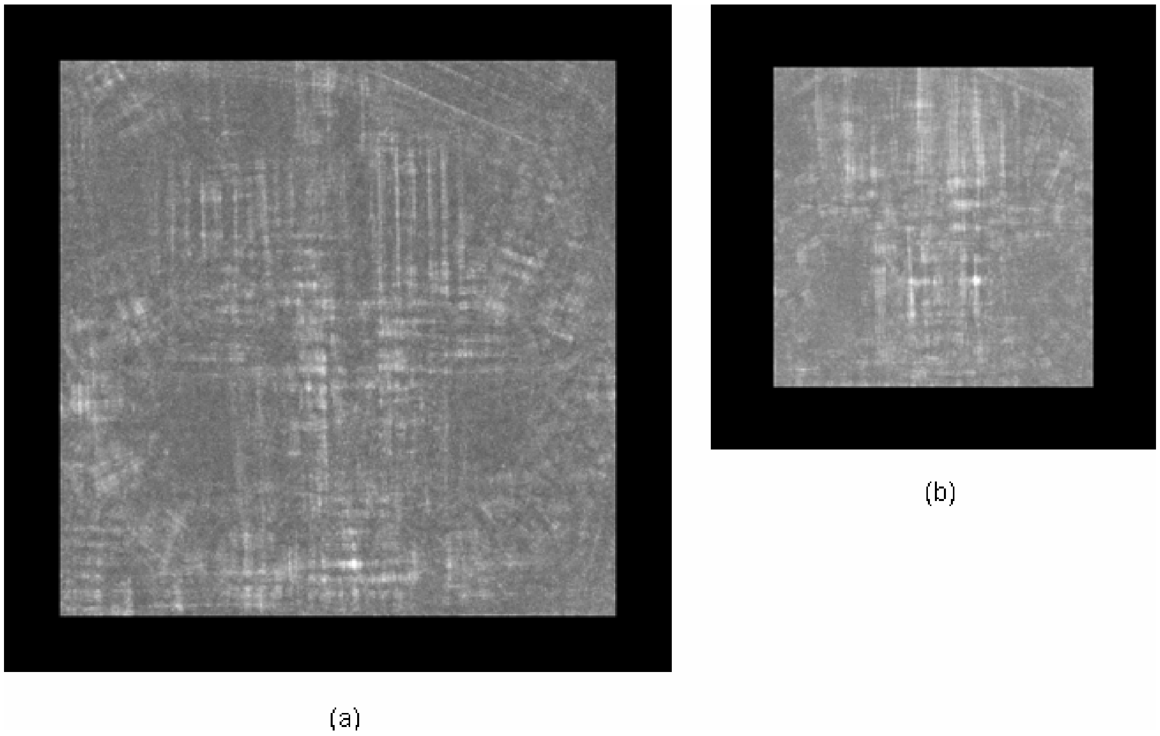
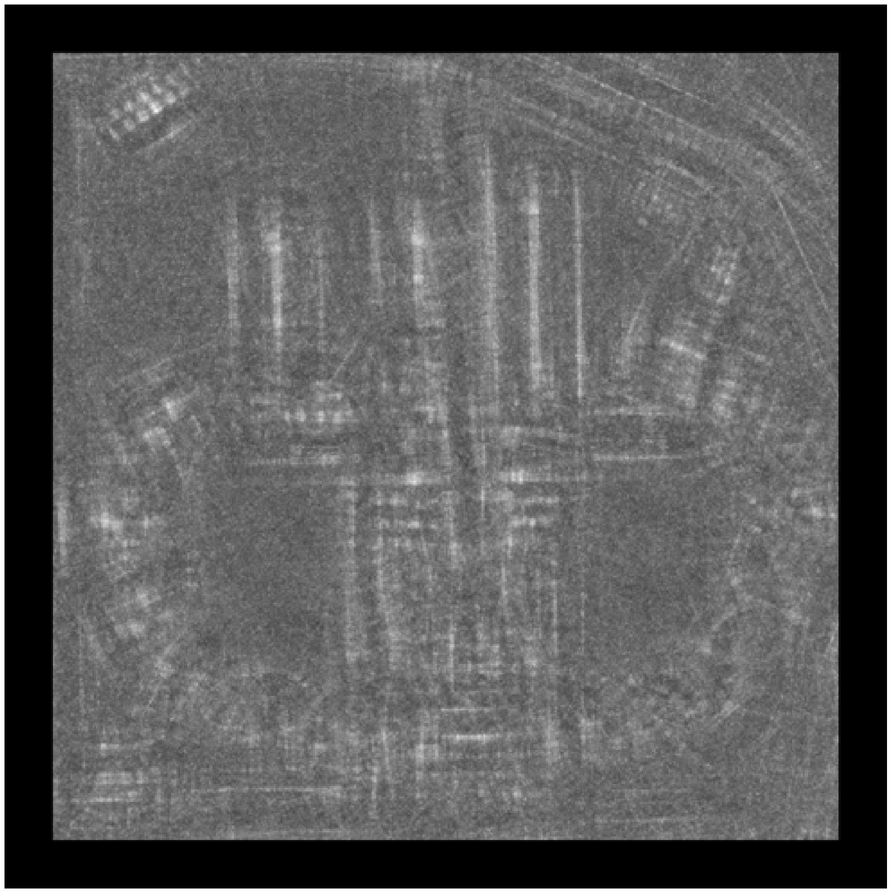


Fig.4.4 Image space model edge projections for three types of buildings in the prison image: (a) notched building (b) long building (c) grated building.



(a) (b)



(c)

Fig.4.5 Surfaces of match similarities (from dark 0 to light 1) for the three models (a)-(c) in Fig.4.4 against the prison image in Fig.4.3.

4.5 Partial Models and Exact vs. Inexact Model Matching

Although phase sensitive matching is primarily intended for finding exact matches to specific objects, it can also be used to find constrained inexact matches to entire families of objects. The ability to find inexact matches to specific models is inherent in phase sensitive matching. For example, if one uses phase sensitive matching to find rectangular courtyard buildings in an image, one can expect to find strong matches not only to courtyard buildings, but also somewhat weaker matches to buildings of roughly the same size that resemble courtyard buildings.

The ability to find matches to entire families of objects is attributable to the fact that objects can be modeled with varying degrees of completeness. *Partial models*, which contain incomplete information about the spatial properties of an object, can be much easier to generate than complete models, and can be used to detect not only specific objects, but entire families of objects that share common spatial attributes. Partial models contain spatial attributes shared by one family of objects. For example, if the goal is to detect buildings that contain, among other things, a wing with specific geometric properties, then a model that partially captures the geometry of only that wing may be all that is required.

5. Match Disambiguation

Match disambiguation is the process of extracting a complete set of unambiguous matches from a surface $S(c,r)$ of match similarities. A match at pixel $[c,r]$ is unambiguous if there is no pixel $[c',r']$ for which $S(c,r) \leq S(c',r')$ and the matches at $[c,r]$ and $[c',r']$ are in conflict. Two criteria for conflicting matches are discussed in this section. In *conflict by proximity*, two matches conflict of their projected edge centroids lie within a prescribed distance (typically in pixels) of each other. Disambiguation by proximity is useful for detecting objects that are expected to occur sparsely. In *conflict by overlap*, two matches conflict if their flattened model projections overlap. Flattened model projections were described in Section 2 as being produced from flattened object space models, i.e., object space models in which z is forced to be constant for all parameters of all graphics primitives that the 3D solid model is composed of. In disambiguation by overlap, flattened model projections prevent confusion between overlap and occlusion. Disambiguation by overlap is useful for detecting objects that can be close together.

The list of disambiguated matches to a prescribed object extracted from a prescribed image can be stored in an output file containing a list of match state (or feature) vectors. Each state or feature vector contains a match position (in pixel coordinates), a match orientation, and a similarity value (from 0 to 1).

5.1 Disambiguation by Proximity

In disambiguation by proximity, a match $S(c,r)$ at pixel $[c,r]$ is unambiguous if and only if $S(c,r) > S(c',r')\ \forall\ [c',r'] \in R(c,r \mid \Delta)$. Δ is some user-specified Euclidean distance, in pixels, and $R(c,r \mid \Delta)$ is the set of all pixels whose centers lie within Δ pixels of the center of pixel $[c,r]$. Efficient algorithms for performing disambiguation by proximity are developed in this section.

Disambiguating a List of Matches by Proximity

When disambiguation by proximity is applied to lists of matches, the lists either come from a single image block or from across multiple image blocks. Lists that come from a single image block are normally extracted directly from the match surface $S(c,r)$ generated for that block, as described later in this section. Lists that come from across multiple image blocks are normally formed by appending lists of matches disambiguated by proximity from successive image blocks. In each case, the lists typically contain matches that are not too densely spaced. Such lists of matches can be efficiently disambiguated by proximity as follows. First, divide the image or image block from which the list was extracted into m rows and n columns of square virtual tiles of width Δ pixels. Generate an mxn array of sub-lists of matches, one per tile. Specifically, for $i = 0 \ldots m\!-\!1$ and $j = 0 \ldots n\!-\!1$, tile $[i,j]$ contains $n_{ij} \geq 0$ matches, $\{[c_{ij}(k), r_{ij}(k), \theta_{ij}(k), S_{ij}(k), status_{ij}(k) = 0]\quad k = 0 \ldots n_{ij}\!-\!1\}$, where $c_{ij}(k)$ and $r_{ij}(k)$ are match centroid column and row coordinates, $\theta_{ij}(k)$ is the orientation of best match, $S_{ij}(k)$ is the degree of match similarity, and $status_{ij}(k)$ is 1 or 0 depending on whether the match is or is not ambiguous. The following algorithm segment efficiently determines the ambiguity status of every match in a list extracted from an image with R rows and C columns of pixels:

```
for $i = 0 \ldots m-1$
    for $j = 0 \ldots n-1$
        for $k = 0 \ldots n_{ij}-1$
            if $status_{ij}(k) = 1$
                continue
            for $i' = max(0, i-1) \ldots min(R-1, i+1)$
                for $j' = max(0, j-1) \ldots min(C-1, j+1)$
                    for $k' = 0 \ldots n_{i'j'}-1$
                        if $\| c_{i'j'}(k') - c_{ij}(k) \|^2 + \| r_{i'j'}(k') - r_{ij}(k) \|^2 > \Delta^2$ or $= 0$
                            continue
                        if $S_{ij}(k) \geq S_{i'j'}(k')$
                            $status_{i'j'}(k') \leftarrow 1$
                        if $S_{ij}(k) \leq S_{i'j'}(k')$
                            $status_{ij}(k) \leftarrow 1$
```

Disambiguating a Surface of Matches by Proximity

Matches disambiguated by proximity within a single image block can be efficiently extracted directly from its match surface $S(c,r)$ as follows:

```
$\Delta_2 \leftarrow max(1, \Delta/2)$,   copy $S$ into $S'$
$\forall\ (c,r)$
    if $S'(c,r) > 0$
        $S_{max} \leftarrow \max_{(c',r') \in R(c,r \mid \Delta_2)} S(c',r')$
        $n_{max} \leftarrow$ number of $(c',r') \in R(c,r \mid \Delta_2)$ : $S(c',r') = S_{max}$
        if $n_{max} > 1$
            $S_{max} \leftarrow 2$
        $\forall\ (c',r') \in R(c,r \mid \Delta_2)$
            if $S(c',r') < S_{max}$
                $S'(c',r') \leftarrow 0$
```

The partially disambiguated matches occur at (c,r) : $S'(c,r) > 0$. Disambiguate the list of partially disambiguated matches by proximity Δ (as described earlier).

In summary, to disambiguate an entire image by proximity, the match surface from each image block is first disambiguated by proximity. This produces one list of disambiguated matches for each image block. These lists are then combined, and the combined list is independently disambiguated by proximity.

5.2 Disambiguating a List of Matches by Overlap

In disambiguation by overlap, a match $S(c,r)$ at pixel $[c,r]$ is unambiguous if and only if $S(c,r) > S(c',r')$ at all $[c',r']$ for which the flattened model projections at $[c,r]$ and $[c',r']$ overlap. Consider two edge projections $P_i = \{[c_i(k), r_i(k)]\ \ k = 0 \ldots n_i-1\}$, $i = 1,2$. By brute force, the process of determining if

P_1 and P_2 overlap has O[$n_1 n_2$] complexity. However, if a bitmap B_1 is first created from P_1, the process can be reduced to O[n_1+n_2] complexity:

```
for  $k = 0 \ldots n_2-1$
    if  $B_1(c_2(k), r_2(k)) = 1$
        return  overlapStatus = $1$
return  overlapStatus = $0$
```

Within a single image block, disambiguation by overlap can be computationally expensive if matches at each pixel on the match surface are compared to matches at every other pixel. The cost can be reduced by recognizing that flattened projections cannot overlap if their centroids are more than $2R_{max}$ pixels apart. The cost can be further reduced by first disambiguating the match surface by a proximity of

(5.1) $$\Delta = max\,(\alpha\bar{R}, \Delta_{min})$$

for some sufficiently small Δ_{min} (say Δ_{min} = 3 pixels). Even for $\Delta = \Delta_{min}$, disambiguation by proximity will eliminate the majority of matches that need to be considered for disambiguation by overlap. Although some objects that lie close to each other might be missed if equation (5.1) is used, at least one of the objects in the area should be unambiguously matched, and only one such object needs to be matched in order for a human analyst cue to be successfully generated for that area.

Across multiple image blocks, disambiguation by overlap will be efficient if the input list of matches has already been disambiguated by proximity. However, R_{max} will no longer be the maximum radius for projections onto a specific image block, as in the case of disambiguation by overlap within a single block. R_{max} can instead be taken as the maximum radius for projections over all image blocks.

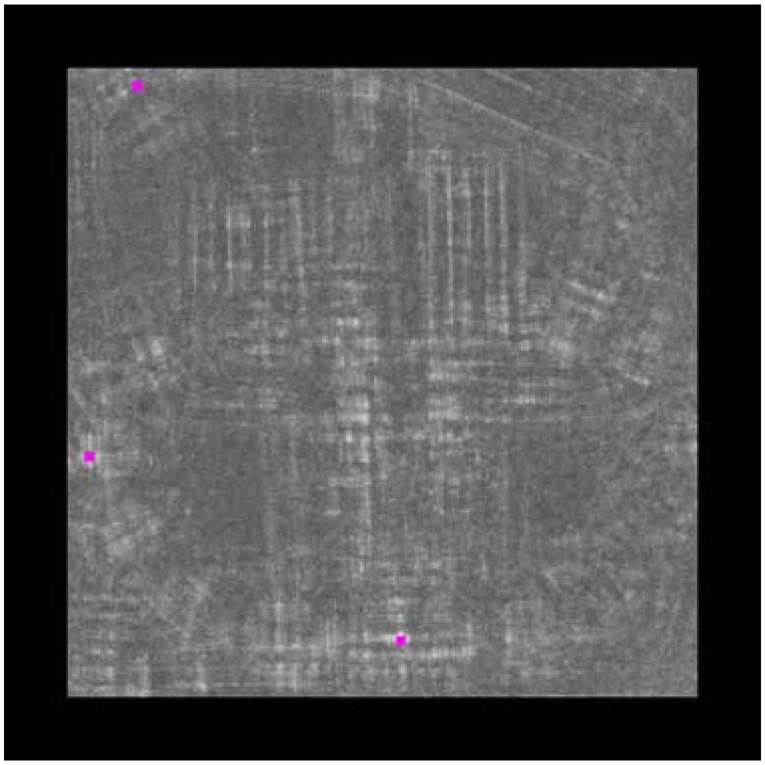
In any case, the process of disambiguating a list of matches by overlap differs only slightly from the process of disambiguating a list of matches by proximity. First, Δ is replaced by $2R_{max}$. Then, an overlap test must be inserted after the proximity test in the inner loop:

```
if  $match_{ij}(k)$  does not overlap $match_{i'j'}(k')$
    continue
```

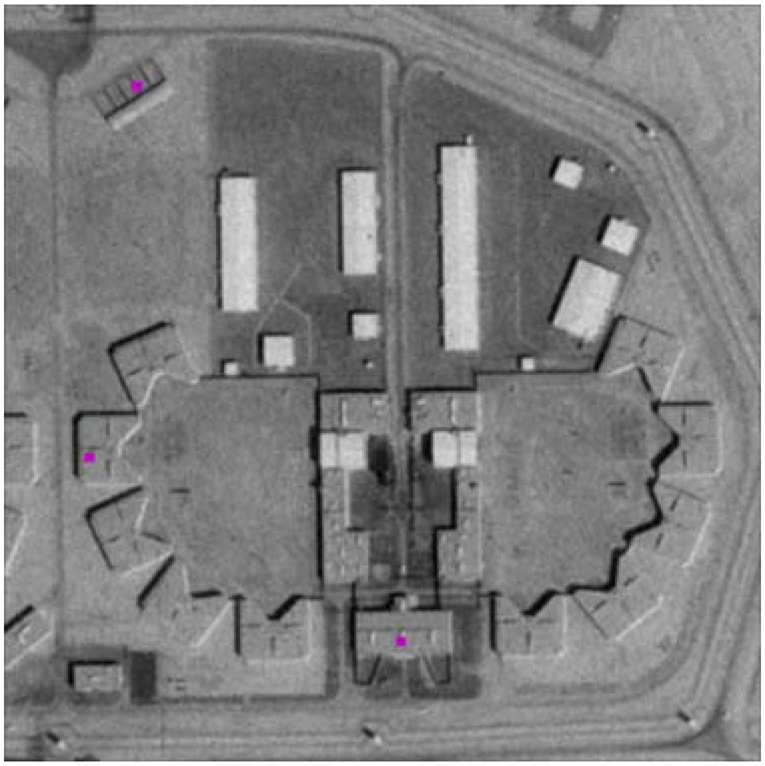
In summary, to disambiguate an entire image by overlap, the match surface from each image block is first disambiguated by a proximity dictated by the mean radius of projections onto that block. This produces one list of disambiguated matches for each image block. Each of these lists is disambiguated by overlap. The resulting lists are then combined, and the combined list is independently disambiguated by overlap.

5.3 Examples

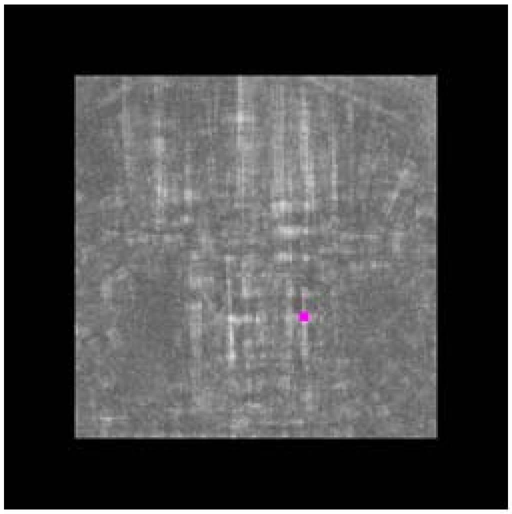
In Fig.5.1-5.3, disambiguated matches are depicted as colored dots overlaid on the match surfaces in Fig.4.5. The matches are disambiguated by proximities of 128 and 32 pixels in Fig.5.1-5.2 respectively, so the images in Fig.5.2 contain more spatially dense distributions of matches than the corresponding images in Fig.5.1. The matches are disambiguated by overlap in Fig.5.3. For reference, the locations of the disambiguated matches are shown overlaid on the image in Fig.4.3 alongside the match surfaces.

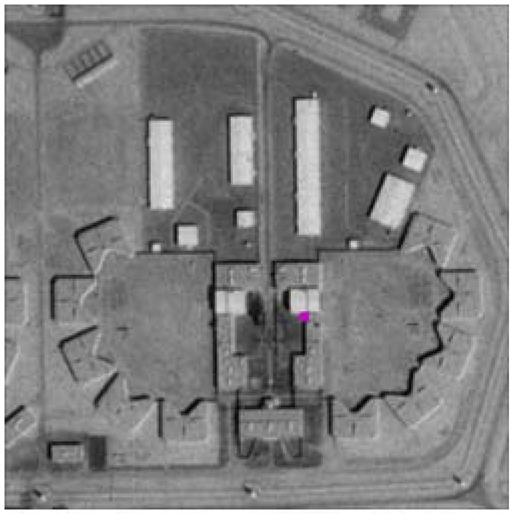


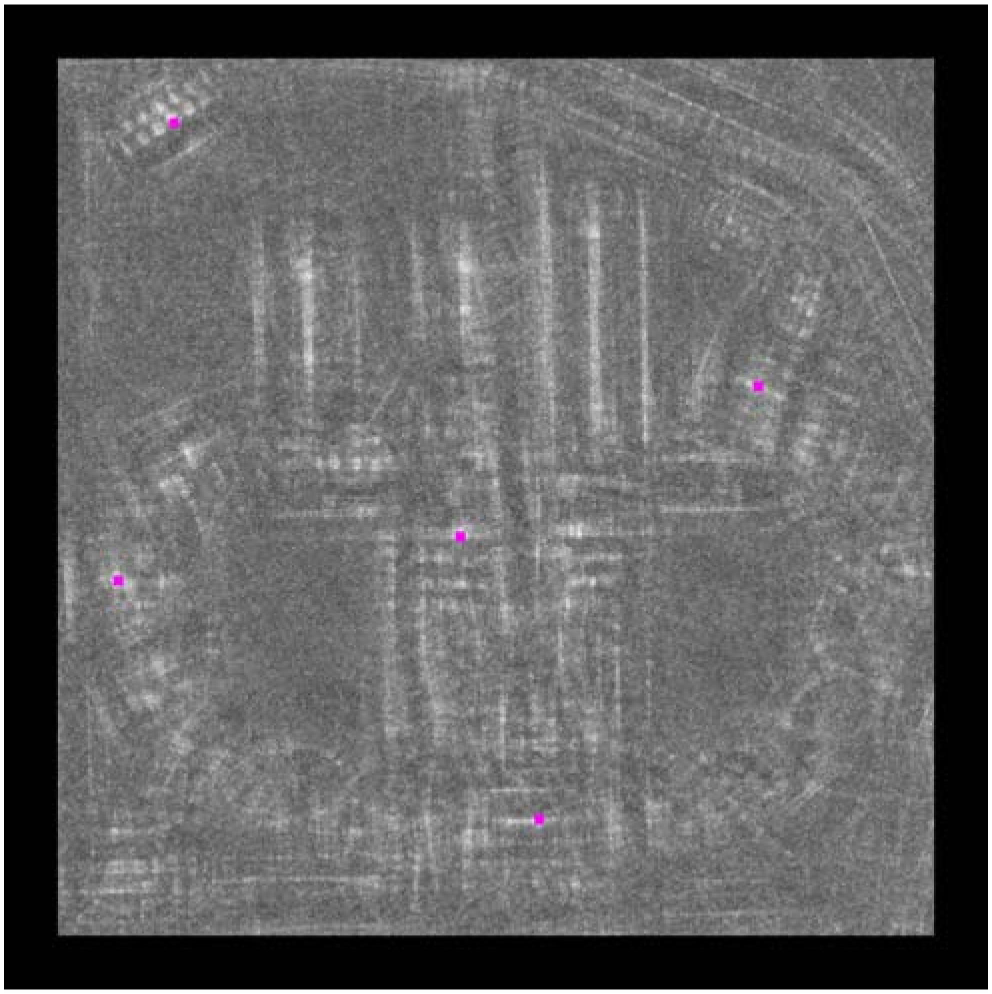
(a)

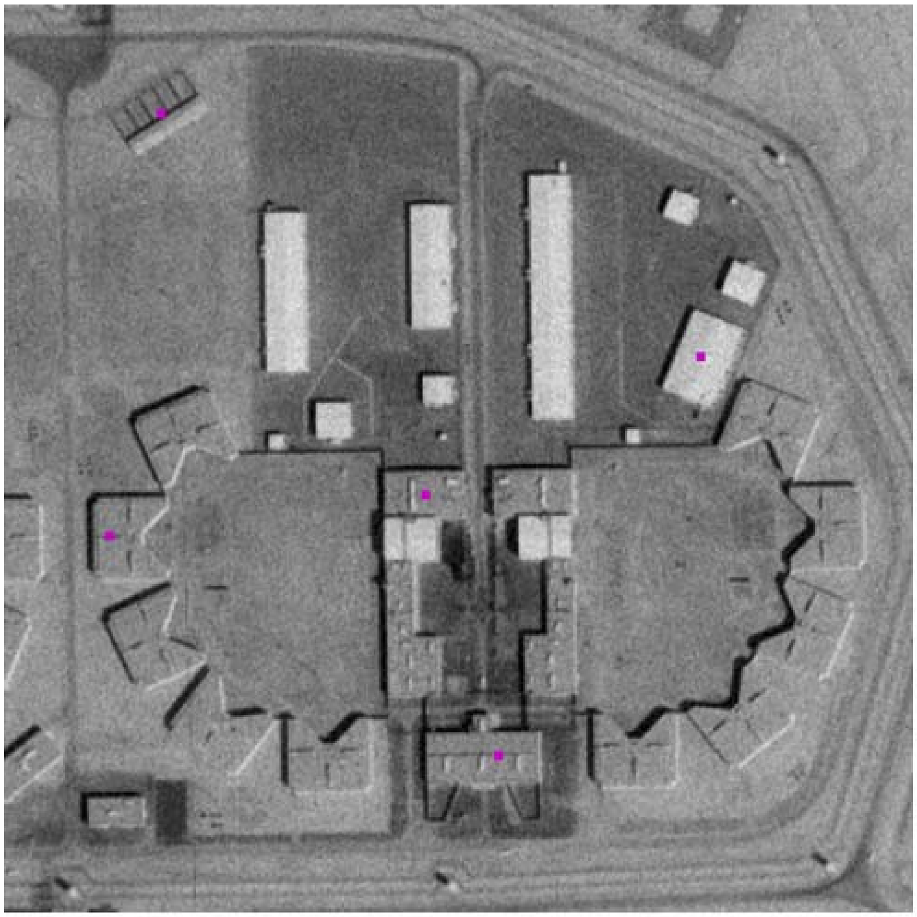


(b)


(c)

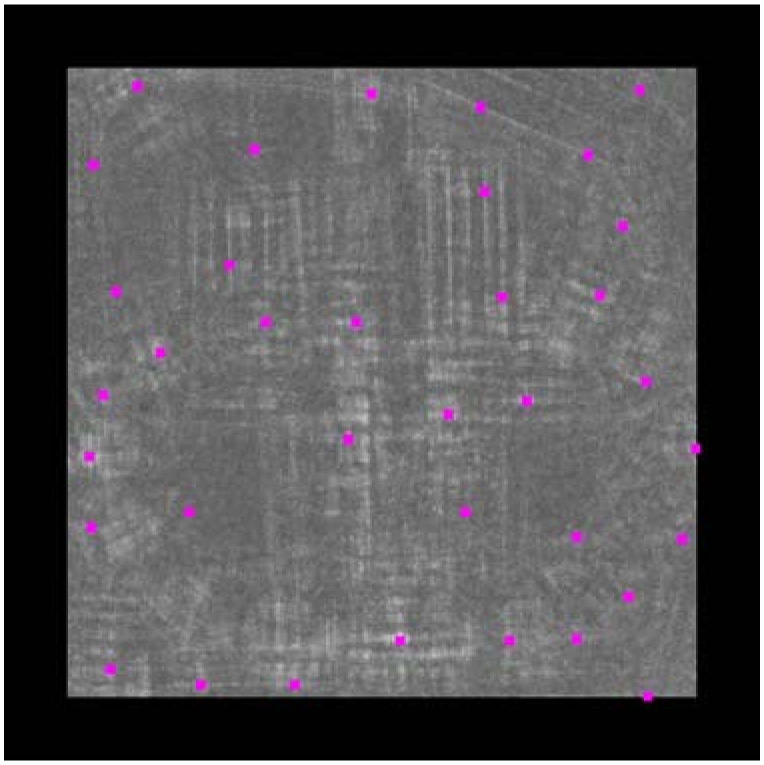

(d)


(e)

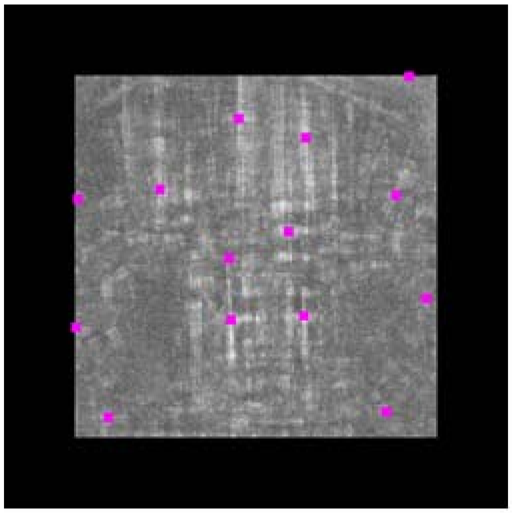


(f)

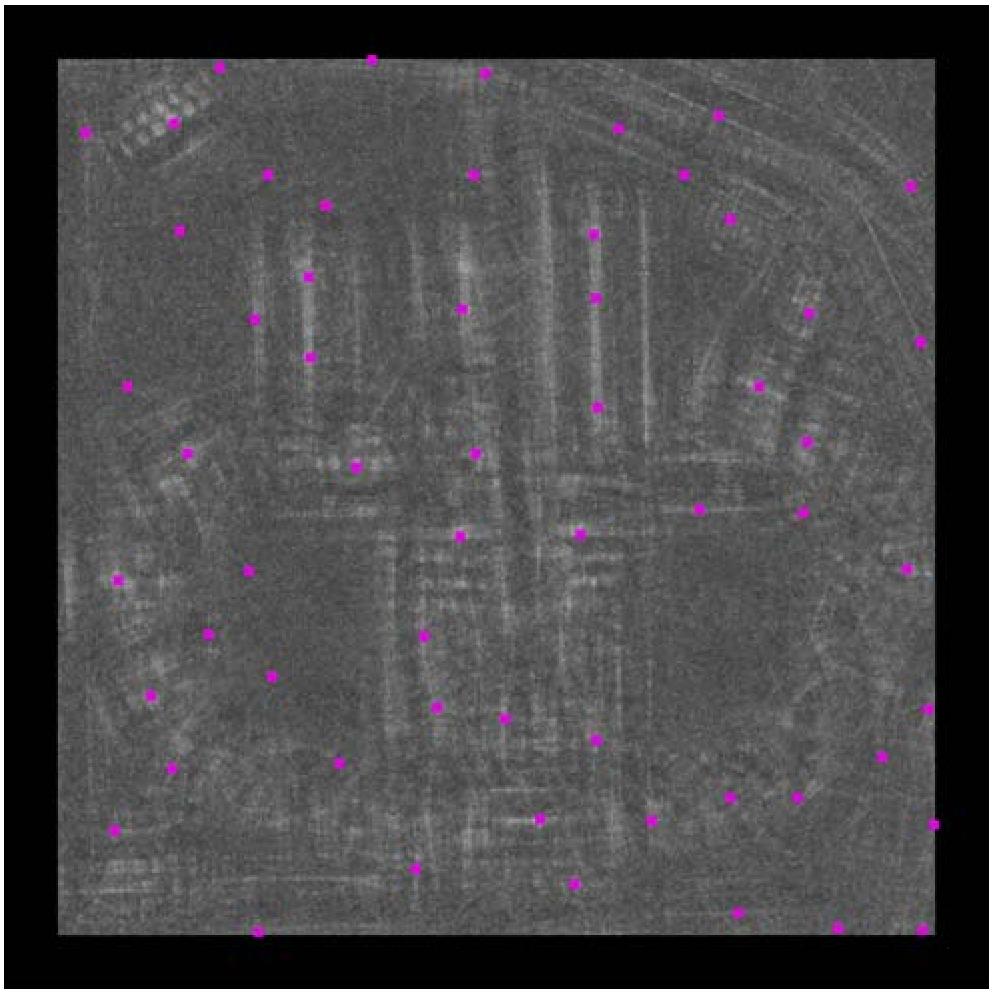
Fig.5.1 Matches disambiguated by a proximity of 128 for the prison image in Fig.4.3 and (a)-(b) notched building (c)-(d) long building (e)-(f) grated building. The unambiguous match positions are shown as colored dots..

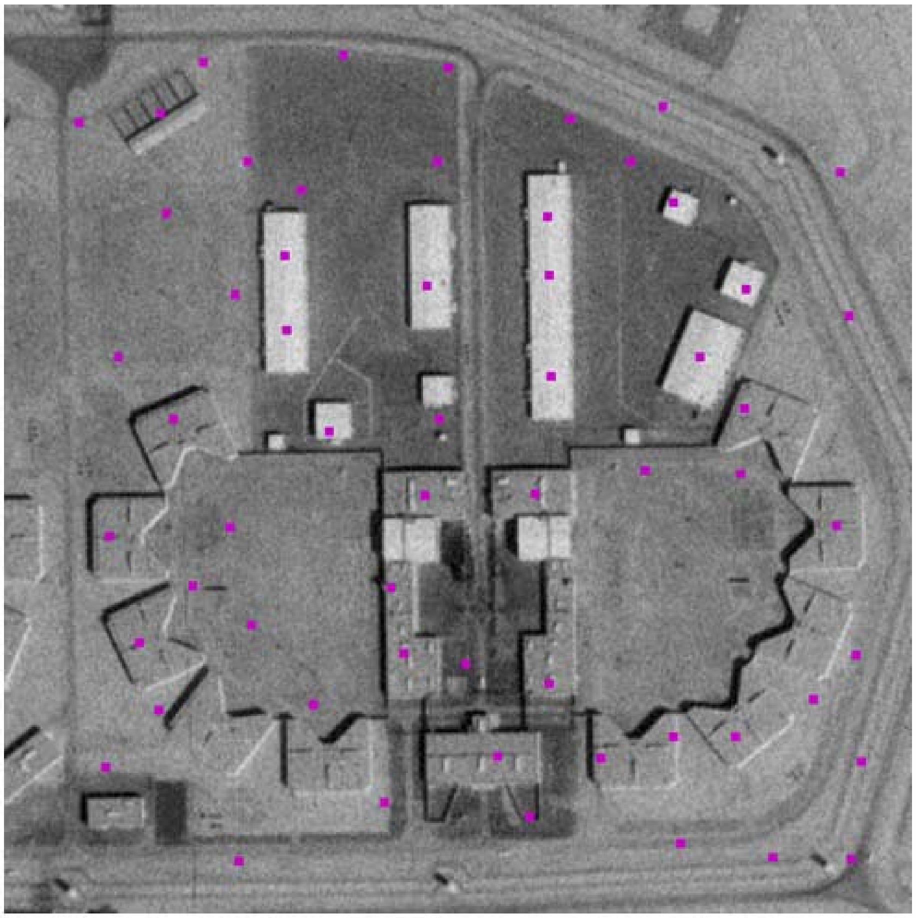

(a)

(b)


(c)

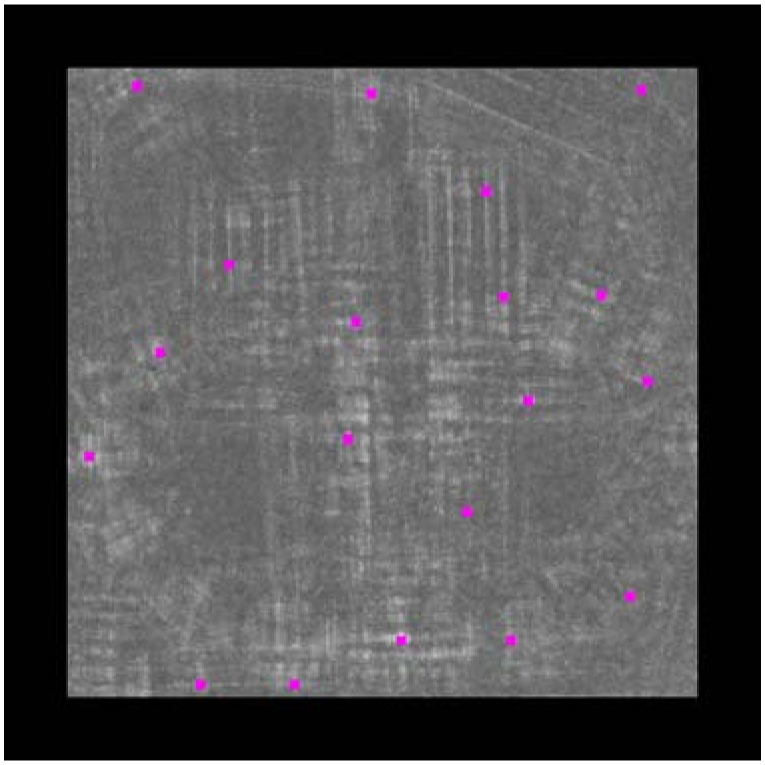
(d)


(e)

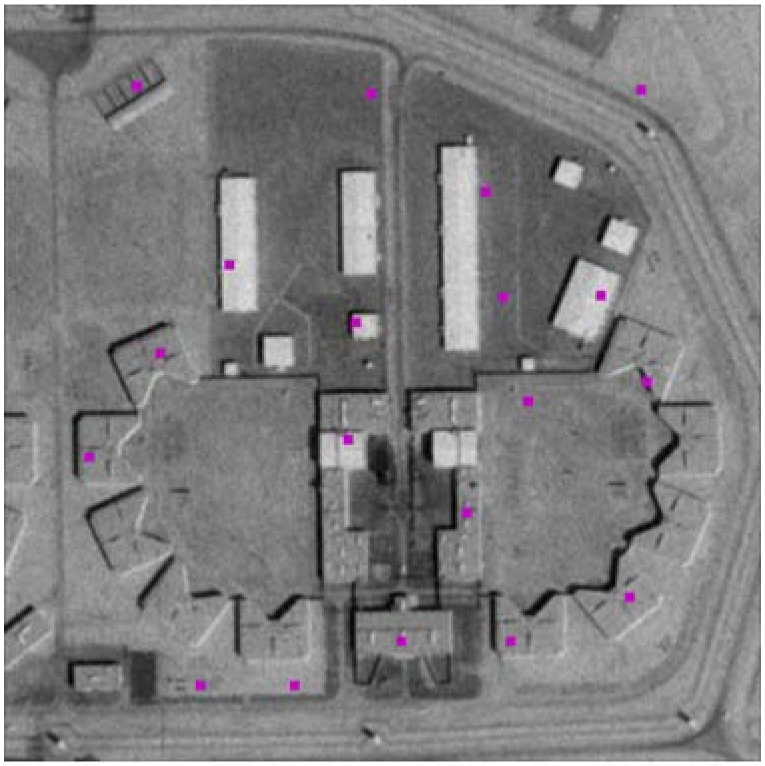


(f)

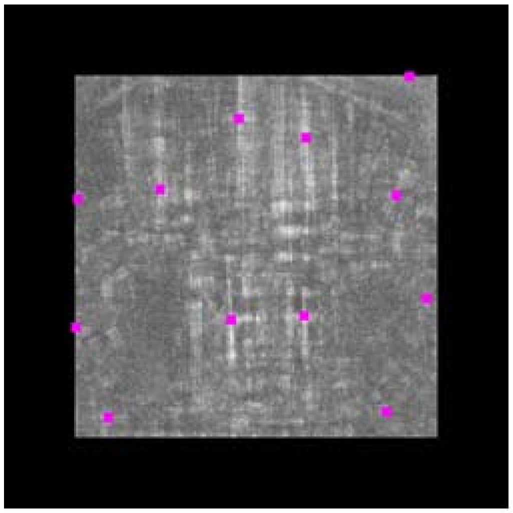
Fig.5.2 Matches disambiguated by a proximity of 32 for the prison image in Fig.4.3 and (a)-(b) notched building (c)-(d) long building (e)-(f) grated building.

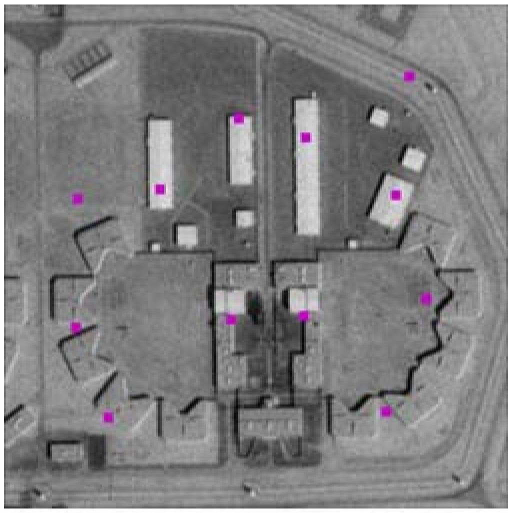


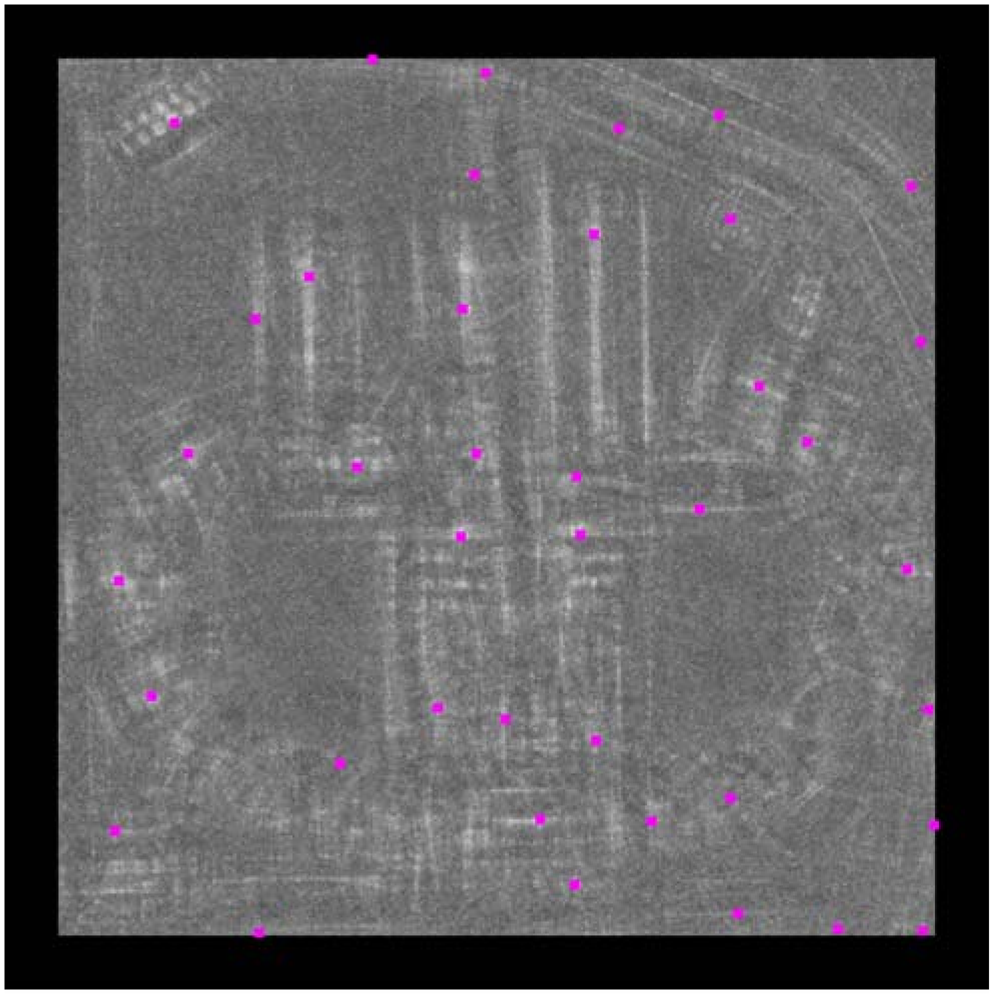
(a)



(b)


(c)


(d)


(e)

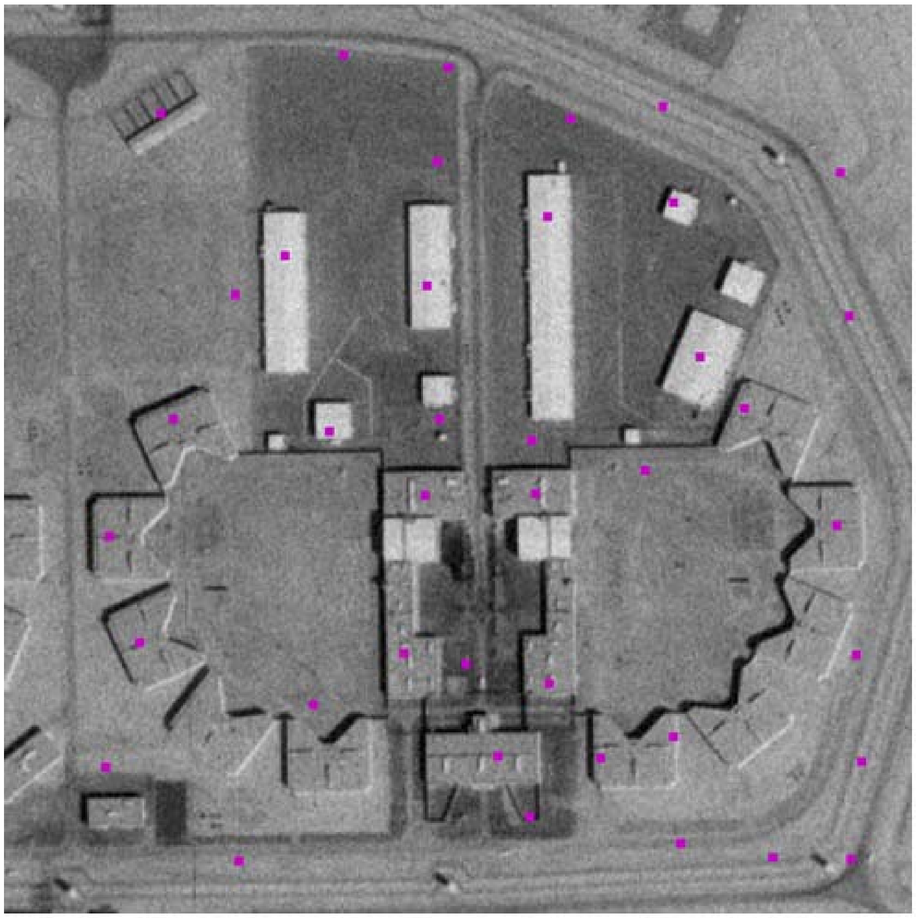

(f)

Fig.5.3 Matches disambiguated by overlap for the prison image in Fig.4.3 and (a)-(b) notched building (c)-(d) long building (e)-(f) grated building.

6. Scalability and Image Block Series Generation

Phase sensitive cueing can be scaled to multi-band images and to images of arbitrary size. As discussed in Section 3.1, scalability to multi-band images can be achieved by generalizing pixel phase estimation in equation (3.2) to multi-band images. Scalability to images of arbitrary size involves generating a series of blocks to be independently processed in succession. Although it is sometimes sufficient to use image blocks of fixed size and overlap Δ, variable block size and overlap are sometimes required. In general, an image block is characterized by the pixel [*column, row*] coordinates $[c_{UL}, r_{UL}]$ of its upper left corner, its width and height $[w, h]$, and its spatial scale factor α. For convenience, assume that $h = w$ for blocks that are not clipped by the image border, in which case, unclipped blocks are taken to be squares of width w pixels.

6.1 Image Block Series Generation for Image Space Model Matching

The spatial extent of edge projections for image space models of objects does not vary as a function of position within an image. In this case, the same set of projections (and projection DFT's depicted as shaded boxes in Fig.4.2) is applied to the entire image, so the projections and their DFT's computed for the first image block do not have to be re-computed for any subsequent image block.

For image space model matching, it is sufficient to use square image blocks of fixed width w and fixed overlap $\Delta < w$. For radix 2 FFT-based phase sensitive matching, the width w of unclipped image blocks is given by equation (4.11) with w_α defined as some power of 2 that satisfies the memory constraint inequality (4.14) (say $w_\alpha = 512$ pixels). The spatial scale factor α in equation (4.11) is derived using equation (4.10) for a specified target mean projection radius of $\bar{R}_T$ (say $\bar{R}_T = 20$ pixels). The overlap Δ (in pixels) between successive rows and columns of image blocks must be large enough to eliminate the possibility that a projection will fail to be matched due to the fact that it spans the border of some image block. In general, the smallest allowable value for Δ is thus

(6.1) $$\Delta = 2R_{max}$$

Image space model projections at all orientations are guaranteed to lie completely within an image block of size h x w if their centroids are at least R_{max} pixels from any block border. Thus, as illustrated in Fig.6.1, matching should only be performed within a matching sub-block of size h' x w', where $h' = max(0, h-2R_{max})$, $w' = max(0, w-2R_{max})$, and the upper left corner of the sub-block occurs at column and row coordinates of R_{max}. If $min(w',h') = 0$, then matching should not be performed because the block is too narrow. Note that the overlap between successive image blocks is specified such that the matching sub-blocks are adjacent and non-overlapping.

For an image with W columns and H rows of pixels, the block on the i^{th} row and j^{th} column of blocks has parameters

(6.2) $$[c_{UL}(i,j), r_{UL}(i,j), w(i,j), h(i,j)] = [j(w-\Delta), i(w-\Delta), min[w, W\text{-}c_{UL}(i,j)], min[w, H\text{-}r_{UL}(i,j)]]$$

The number of columns and rows of image blocks is given by

(6.3) $$[N_C, N_R] = \left[n_C + int\left(\frac{W}{w-\Delta}\right), n_R + int\left(\frac{H}{w-\Delta}\right)\right]$$

where $n_C = 0/1$ if W is/is not a multiple of $w-\Delta$, and $n_R = 0/1$ if H is/is not a multiple of $w-\Delta$.

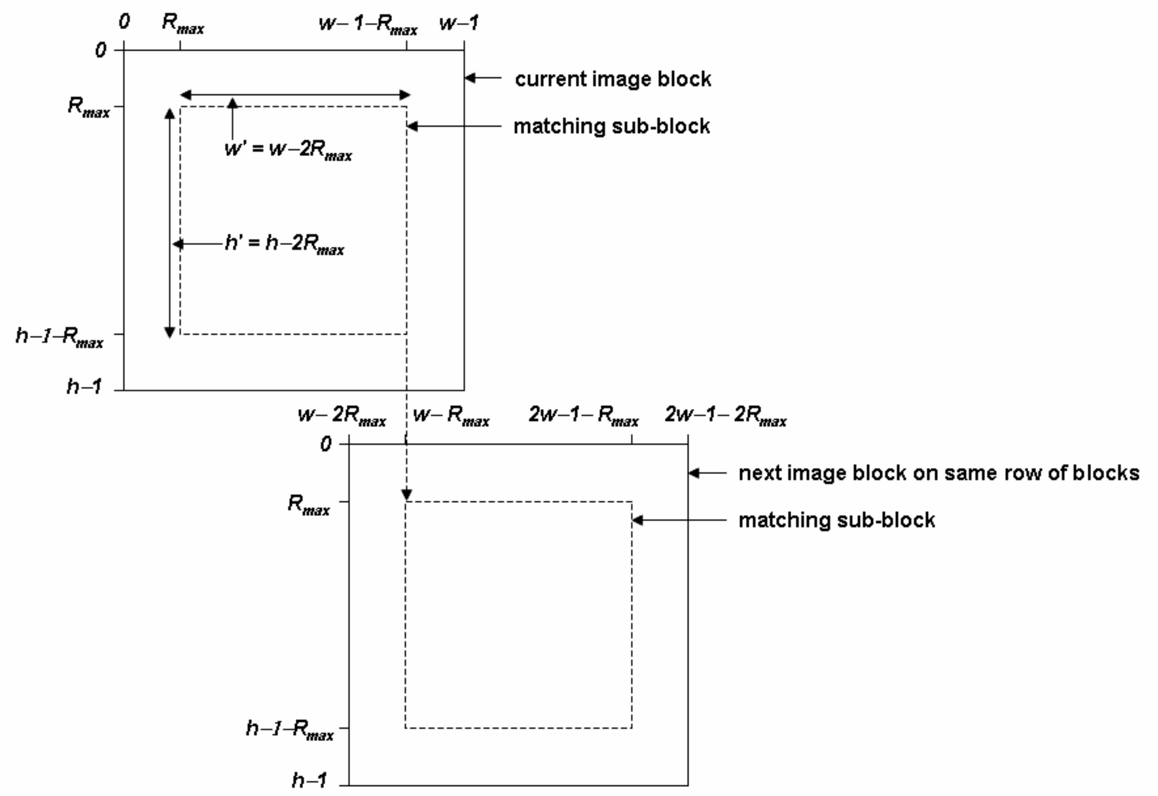


Fig.6.1 Image block series geometry for blocks of fixed size and overlap.

6.2 Image Block Series Generation for Object Space Model Matching

The spatial extent of edge projections for object space models can vary as a function of position within an image. In this case, it is therefore necessary to use image blocks of variable width and overlap that collectively cover the entire image. For the case of variable block size and overlap, it will be helpful to extend the image block parameters for block (i,j) to include the following:

$[c_{UL}(i,j), r_{UL}(i,j)]$: block upper left corner column and row
$[c_0(i,j), r_0(i,j)]$: block center column and row
$[w(i,j), h(i,j)]$: block width and height
$\alpha(i,j)$: block scale factor
$reproject(i,j)$: 1/0 $\Rightarrow$ object does/does not have to be newly projected

When $reproject(i,j) = 0$, the projections (and thus the projection DFT's depicted as shaded boxes in Fig.4.2) do not have to be re-computed for block (i,j). Instead, the projections and projection DFT's applied to the previous block can be applied to block (i,j). If an image can be accurately modeled as a parallel projection (i.e., if the effective lines-of-sight through each pixel are all nearly parallel, as in the case of an EO imaging sensor with a very long effective focal length), the projections and their DFT's will often only need to be computed once over the entire image, since the projections will tend to be mostly invariant to location in the image.

For object space models, the first step in generating a series of image blocks is to estimate the degree of variation in projection size across the image. This can be done by computing an estimate $\hat{R}_{max}$ of R_{max} for projections centered on evenly spaced pixels starting from $[0,0]$ and separated by w_α (again, say $w_\alpha = 512$). Maximum projection radii can be efficiently estimated with bounding cylinders, as described in Section 2.4. Determine the coordinates $[c_{min}, r_{min}]$ and $[c_{max}, r_{max}]$ of the evenly spaced pixels for which $\hat{R}_{max}$ is smallest and largest respectively. Then compute the actual maximum radius $R_{max,0}$ from a conservative number of orientations (say $N = 36$ orientations in 10° increments) for projections centered on $[c_{max}, r_{max}]$. Also, compute the actual mean radius $\bar{R}_0$ from the same number of orientations for projections centered on $[c_{min}, r_{min}]$. An upper bound on the required block overlap is estimated as $\Delta = 2R_{max,0}$. A lower bound on required block width is estimated from $\bar{R}_0$ using equation (4.10) to compute the image spatial scale factor α and then equation (4.11) to compute w from w_α (to be extra conservative, it is useful to reduce $\bar{R}_0$ by replacing $\bar{R}_0$ with the maximum of $f\bar{R}_0$ and $\bar{R}_0 - \Delta_R$ for say $f = 0.95$ and $\Delta_r = 2$ pixels). If $\Delta \geq w$, there is too much variation in projection size across the image, in which case the image should first be divided into sub-images. Otherwise, a series of block centers $[c_0(i,j), r_0(i,j)]$ can be generated as

(6.4) $$[c_0(i,j), r_0(i,j)] = [j(w-\Delta) + w(i,j)/2,\ i(w-\Delta) + h(i,j)/2]$$

where $w(i,j)$ and $h(i,j)$ are computed using equation (6.2) and $i = 0 \ldots N_R-1$; $j = 0 \ldots N_C-1$ with N_R and N_C specified in equation (6.3).

The dimensions of the image blocks in the series specified by equation (6.4) will vary depending on how the projection spatial extents vary across the image:

```
for block $[0,0]$
    · compute $[\hat{R}_{max}, \bar{R}, \alpha, w]$
    · $[\hat{R}_{max,0}, \bar{R}_0, \alpha_0, w_0, reproject] \leftarrow [\hat{R}_{max}, \bar{R}, \alpha, w, 1]$

for blocks $[i,j]$ : $i = 0 \ldots N_R-1$ , $j = 0 \ldots N_C-1$ , $[i,j] \neq [0,0]$
    · compute $\hat{R}_{max}$
    · if $|\hat{R}_{max} - \hat{R}_{max,0}| \leq \Delta_R$ (say $\approx$ 5 pixels)
          $[\alpha, w, reproject] \leftarrow [\alpha_0, w_0, 0]$
      else
          · compute $[\bar{R}, \alpha, w]$
          · $[\hat{R}_{max,0}, \bar{R}_0, \alpha_0, w_0, reproject] \leftarrow [\hat{R}_{max}, \bar{R}, \alpha, w, 1]$

$\forall$ blocks $[i,j]$
    · $[c_{UL}, r_{UL}] \leftarrow [c_0 - w/2, r_0 - w/2]$
    · $w_0 \leftarrow w$ ,  $[w, h] \leftarrow [\min(w_0, W - c_{UL}), \min(w_0, H - r_{UL})]$
```

6.3 Image Block Series Generation for Constrained Searches

Although it is sometimes necessary to perform unconstrained searches over entire images for objects of interest, it is also sometimes possible to constrain the searches to specific portions of the image. Search time can be significantly reduced if the search can be constrained to relatively small portions of the image. The search constraints are often based on information from a non-image source (such as a Geographical Information System or GIS), or from historical information derived from previous images. For example, when searching an image for railcars, it is helpful to have access to a GIS that contains railroad network geo-coordinates.

An image block series for a constrained search can be viewed as a subset of an image block series for an unconstrained search. An image block series for a constrained search can be established by projecting the search constraint geo-coordinates onto the image (as in Section 2.1). If the search constraint geo-coordinates are polygon or polyline vertices, then all image pixels within the polygon or along the polyline can be flagged. Then for constrained search, only image blocks for unconstrained search that contain flagged pixels need to be processed.

7. Phase Sensitive Cueing

In large-scale image analysis, one must specify the tasks to be performed by computers as opposed to human analysts. Although computers are able to quickly extract low and even higher-level features from images, they often cannot easily be programmed to interpret the features correctly. Conversely, human analysts are good at image interpretation, but they cannot quickly cope with large amounts of imagery.

The three basic approaches to large-scale image analysis are manual, automated, and computer-assisted. In manual analysis, a human analyst will typically run image analysis application software on a desktop computer workstation. The human analyst specifies the order in which the images are manually searched. This approach will not be successful if human analysts are unable to meet the analysis time constraints.

Automated analysis is performed without human interaction. Computers are expected to not only extract, but also interpret image features. This approach is not viable if humans lack confidence in the interpretation algorithms. Although robust algorithms exist for certain interpretation tasks, many important interpretation tasks cannot be adequately addressed by existing algorithms. For example, tasks that require specific objects to be detected in images acquired by different types of sensors at various resolutions and under various conditions are not handled robustly enough by current algorithms.

Computer-assisted analysis is a hybrid approach in which computers are expected to extract low and potentially higher-level features from images (as in automated analysis), and human analysts are expected to interpret the results (as in manual analysis). This approach can work in situations where both fully manual and fully automated analysis fail, as long as computers are able to use the extracted features to provide cues that focus human analyst attention on appropriate locations.

This section treats phase sensitive matching in the context of automated vs. computer-assisted analysis. In the automated approach, computers are expected to perform phase sensitive detection of objects which, as demonstrated in Section 7.1, is highly problematic without a human in the loop. However, in the computer-assisted approach, computers are only expected to perform phase sensitive matching followed by match disambiguation. As described in Section 7.2, contiguous image thumbnail tiles must be automatically sorted by some figure-of-merit derived from the disambiguated matches they contain. A human analyst is then expected to interpret the sorted thumbnail cues. The capabilities and limitations of a computer-assisted phase sensitive cueing system are discussed in Section 7.3.

7.1 Problems with Automated Phase Sensitive Detection

Fig.7.1 shows the strongest phase sensitive matches to the three models in Fig.4.4 for a 1024x1024 section of an image of a prison in Calipatria, CA (courtesy of TerraServerUSA). Note that all buildings of each type were detected (the two apparently undetected long buildings are actually mirror images of the model and should not be detected). The image was processed block-by-block in three passes (one pass per object model), as described in Section 6. DFT's of size w_α x w_α were generated for w_α = 512. Each pass used a different series of image blocks. Table 7.1 lists the image block series parameters for the notched building pass.

Table 7.1

Image block series parameters for notched building model and the image in Fig.7.1.

block indices (i ,j)	c_{UL}	r_{UL}	w	h	c_0	r_0	α	*re-project*
(0,0)	0	0	666	666	333	333	.769	1
(0,1)	580	0	444	666	802	333	.769	0
(1,0)	0	580	666	444	333	802	.769	0
(1,1)	580	580	444	444	802	802	.769	0

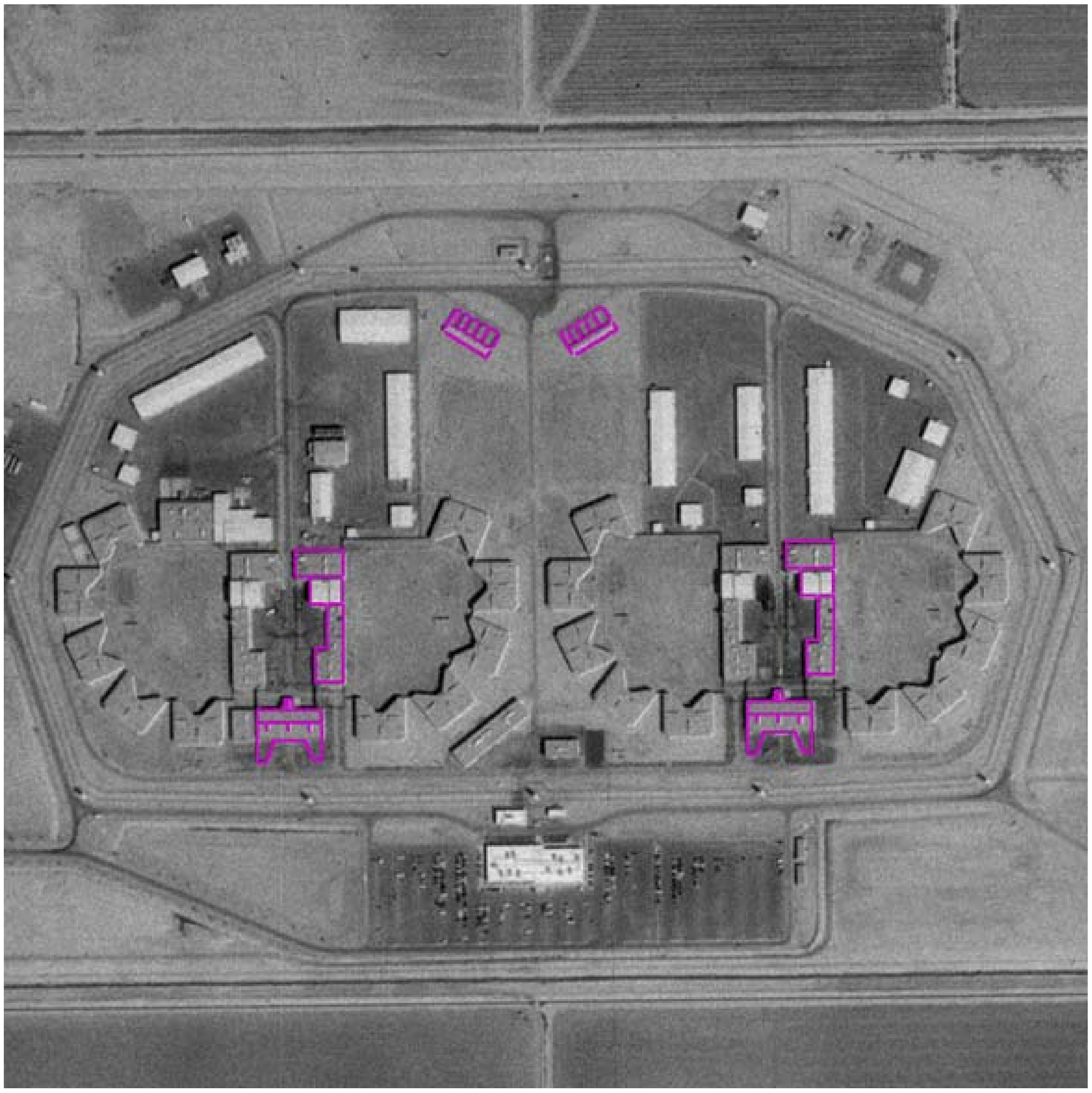


Fig.7.1 Renderings of the strongest phase sensitive matches to the three models in Fig.4.4 for a 1024x1024 section of a prison image.

The results in Fig.7.1 were obtained by applying a decision rule of form

$$S(c,r) \underset{H_1}{\overset{H_0}{\lesseqgtr}} S_0 \in [0,1] \tag{7.1}$$

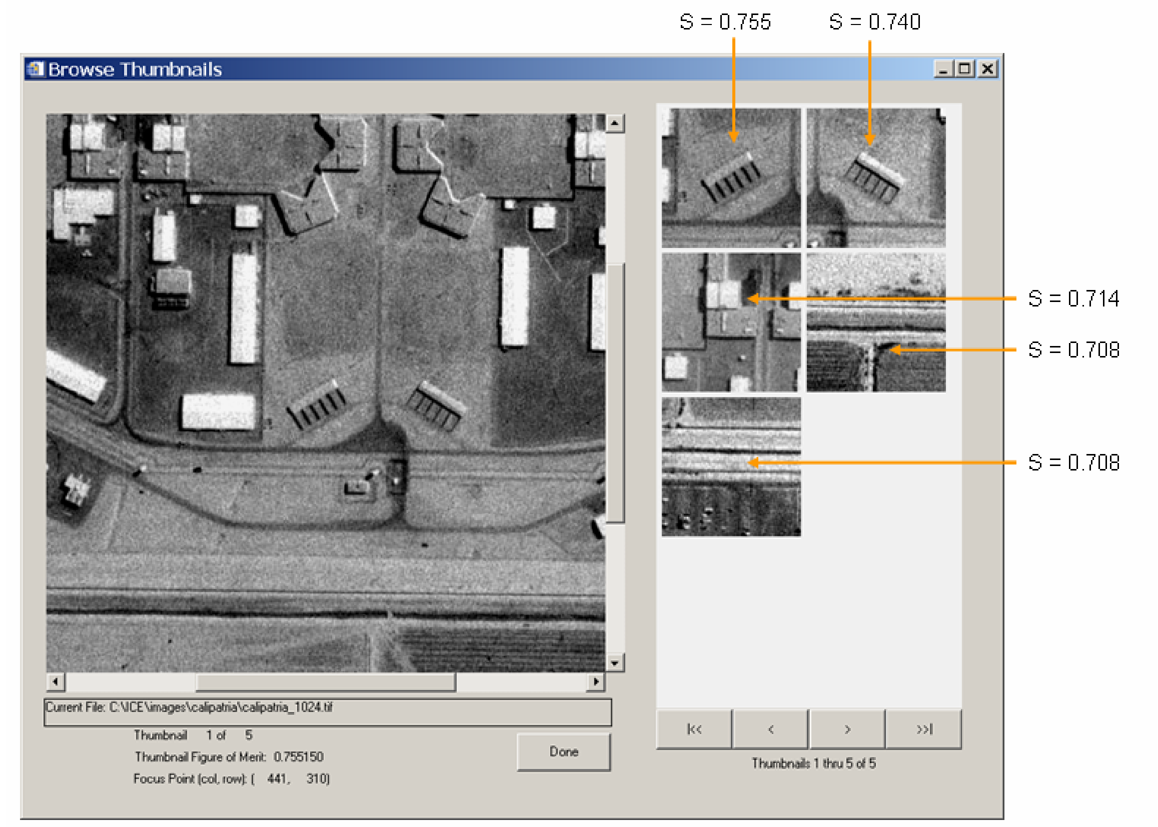
to the locations $[c,r]$ of disambiguated matches, where the null hypothesis H_0 is the "object not detected" hypothesis, the alternative hypothesis H_1 is the "object detected" hypothesis, and S_0 is the decision threshold (in this case, a *similarity threshold*) for object detection. It is clear from Fig.7.1 that there exists a set of decision thresholds for which the detection probability is 1 and the false alarm rate is 0. The similarity thresholds S_0 that produced the results in Fig.7.1 were 0.8, 0.82 and 0.72 for the notched, long and grated buildings respectively. The problem with automated phase sensitive detection is that for any given image and object model, one cannot know a priori what similarity thresholds to use. A human-interactive mechanism is needed.

7.2 Image Thumbnail Cues

One way to address the problem of similarity threshold specification in equation (7.1) is to divide the set of images to be searched into contiguous image thumbnail tiles of fixed size, compute a figure-of-merit for each thumbnail based on the disambiguated matches that it contains, and sort the thumbnails in descending order of figure-of-merit. A human analyst can then visually inspect and interpret thumbnails near the beginning of the list. In this context, image thumbnails near the top of the list are cues that serve to focus the attention of human analysts on specific locations in the set of images to be searched. These thumbnails are presumably the most likely to contain specific objects of interest.

The set of images to be searched can contain large numbers of large images. Even if the set is large, it may take only a matter of seconds or minutes for the system to return a sorted list of thumbnails because phase sensitive matching and match disambiguation will presumably have been automatically performed in advance, and simple hypothesis testing is straightforward once disambiguated matches are available. Although parallel processing may be needed to meet the time constraints for object matching in large sets of images, image thumbnail cueing should only require a single processor. The time needed for a human analyst to inspect image thumbnails near the top of the list should then be very much less than the amount of time needed to inspect the set of images in its entirety.

Fig.7.2 shows a display generated by an image thumbnail cueing application for the 1024x1024 prison image in Fig.7.1 and the grated building model in Fig.4.4(c). The thumbnails on the right are sorted in row-wise descending order of figure-of-merit. Notice that the first two thumbnails contain the two grated buildings that actually occur in the image. When a human analyst clicks on a thumbnail, the surrounding context for that thumbnail is displayed in the larger window on the left. The context for the first thumbnail is displayed in Fig.7.2. The number of thumbnails in the list is limited by a user-specified lower bound on figure-of-merit for image thumbnail cues.



Current File: C:\ICE\images\calipatria\calipatria_1024.tif
Thumbnail 1 of 5
Thumbnail Figure of Merit: 0.755150
Focus Point (col, row): (441, 310)
Thumbnails 1 thru 5 of 5

Fig.7.2 Thumbnail cue results for the grated building in Fig.4.4(c) against the 1024x1024 prison image in Fig.7.1.

The user should be able to specify not only thumbnail size, but also the figure-of-merit for thumbnail sorting (i.e., the thumbnail sort criterion). Four proposed thumbnail figures-of-merit are the maximum, mean, population, and sum of similarities over the strongest so many disambiguated matches that lie within a thumbnail. The first two criteria rate thumbnails based on the best and average quality match contained within the thumbnail. The last two criteria rate thumbnails based more on the number of matches they contain.

Notice that a grated building occurs at the center of the first two thumbnails in Fig.7.2. It is easy for a human analyst to miss a target when it just grazes a highly ranked thumbnail. It is thus important to spatially shift each thumbnail so that its center coincides with a *focus point*. The focus point can be defined as the location of the disambiguated match of largest similarity value within a thumbnail (as in Fig.7.2).

Also, human analysts can easily miss targets in thumbnails that are too dark or that have poor contrast. It is thus important for each thumbnail to be independently subjected to brightness-contrast enhancement before being displayed. The brightness-contrast settings used to enhance a thumbnail should also be used to enhance its context image so that the target of interest, if present, will be equally visible in both.

Finally, it is usually safe to generate a sorted list of thumbnails using thumbnails from a single image or from a set of closely related images. However, in diverse images (e.g., images acquired with different sensors, with different obliqueness, at different times of the day, in different seasons of the year, at different spatial resolutions, etc.), match similarities to a specific object model can vary significantly. It can be risky to generate a sorted list of thumbnails using thumbnails from multiple diverse images if the thumbnail figure-of-merit is sensitive to image diversity. Unless thumbnail figures-of-merit can somehow be "normalized" across images, it is wise to sort and inspect thumbnails separately for each image. However, cueing efficiency across large sets of images can be improved by using a "normalized" figure-of-merit that is relatively insensitive to image diversity. For example, maximum and mean similarity measures are sensitive to image diversity, whereas population measures are less sensitive. Unfortunately, population measures are a poor choice for discrete target cueing. In discrete target cueing applications, a better option might be to design figures-of-merit based on relationships between matches to one or more types of objects in a scene, multiple types of information from images (e.g., spatial and spectral), or information from both image and non-image sources (e.g., image and GIS sources). Developing figures-of-merit that are relatively insensitive to image source is an important challenge for future research.

References

Barrow1977 H. G. Barrow, J. M. Tenenbaum, R. C. Bolles , H. C. Wolf, “Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching”, Proc. 5th Int. Joint Conf. Artif. Intell., Cambridge, MA, 1977, pp.659-663.

Borgefors1988 G. Borgefors, “Hierarchical Chamfer Matching: a Parametric Edge Matching Algorithm”, IEEE Trans. PAMI, Vol.10, No.6, 1988, pp.849-865.

Brown1992 L. G. Brown, “A Survey of Image Registration Techniques”, ACM Comput. Surveys, Vol.24, 1992, pp.325-376.

Canny1986 J. Canny, “Computational Approach to Edge Detection”, IEEE Trans. PAMI, Vol.8, No.6, November 1986, pp.679-698.

Danielsson1980 P. E. Danielsson, “Euclidean Distance Mapping”, CGIP, Vol.14, 1980, pp.227-248.

Davis1975 L. S. Davis, “A Survey of Edge Detection Techniques”, CGIP, Vol.4, 1975, pp.248-270.

DeCastro1987 F. DeCastro and C. Morandi, “Registration of Translated and Rotated Images Using Finite Fourier Transforms”, IEEE Trans. PAMI, Vol.9, No. 5, September 1987, pp.700-703.

Eppler2000 W. G. Eppler, D. W. Paglieroni, S. M. Petersen, M. J. Louie, "Fast Normalized Cross-Correlation of Complex Gradients for Autoregistration of Multi-Source Imagery", Proc. ASPRS DC 2000 Conf., May 22-27, 2000.

Eppler2001 W. G. Eppler, B. Trusso, “System for Registering Site Models to Gray-Scale Images”, unpublished white paper produced under contract NIMA NMA 201-01-C-0013, AFE/CD Topic 2, Autonomous Image to Model Registration, December 2001.

Eppler2003 W. G. Eppler, D. W. Paglieroni, S. M. Petersen, M. J. Louie, "Normalized Crosscorrelation of Complex Gradients (NCCCG) for Image Autoregistration", U.S. Patent 6,519,372, issued February 11, 2003, assignee: Lockheed Martin.

Fonseca1996 L. Fonseca, B. S. Manjunath, ‘Registration Techniques for Multisensor Remotely-Sensed Imagery”, Photogram. Eng. Remote Sensing, Vol.62, 1996, pp.1049-1056.

Grant2004 C. Grant, Iterative Method for Estimating Pixel Geo-Coordinates using Rational Polynomials, Personal Communication, July 2004.

Kuglin1975 C. D. Kuglin, D. C. Hines, “The Phase Correlation Image Alignment Method”, Proc. IEEE 1975 Int. Conf. Cybernetics and Society, IEEE, New York, September 1975, pp.163-165.

Kumar1992 B. V. K. Kumar, F. Dickey, J. DeLaurentis, “Correlation Filters Minimizing Peak Location Errors”, J. Opt. Soc. A., Vol.0, No.5, May 1992.

Marr1980 D. Marr, E. Hildreth, “Theory of Edge Detection”, Proc. Roy. Soc. London, B207, 1980, pp.187-217.

Paglieroni1992 D. W. Paglieroni, “A Unified Distance Transform Algorithm and Architecture”, Machine Vision and Applications, vol.5, 1992, pp.47-55.

Paglieroni1994 D. W. Paglieroni, G. E. Ford, E. M. Tsujimoto, “The Position-Orientation Masking Approach to Parametric Search for Template Matching”, IEEE Trans. PAMI, Vol.16, No.7, July 1994, pp.740-747.

Prewitt1970 J. M. S. Prewitt, “Object Enhancement and Extraction”, in Picture Processing and Pyschopictorics, B. S. Lipkin and A. Rosenfeld, eds., Academic Press, New York, 1970.

Rosenfeld1966 A. Rosenfeld, J. L. Pfaltz, “Sequential Operations in Digital Picture Processing”, J. Assoc. Comput. Mach., Vol.13, 1966, pp.471-494.

Svedlow1976	M. Svedlow, C. D. McGillem and P. E. Anuta, “Experimental Examination of Similarity Measures and Pre-Processing Methods used for Image Registration”, Symposium on Machine Processing of Remotely Sensed Data, Westville, Indiana, June 1976, pp.4A-9.
Yamada1984	H. Yamada, “Complete Euclidean Distance Transform by Parallel Operation”, Proc. 7th Int. Conf. Pattern Recog., Montreal, Canada, 1984, pp.69-71.